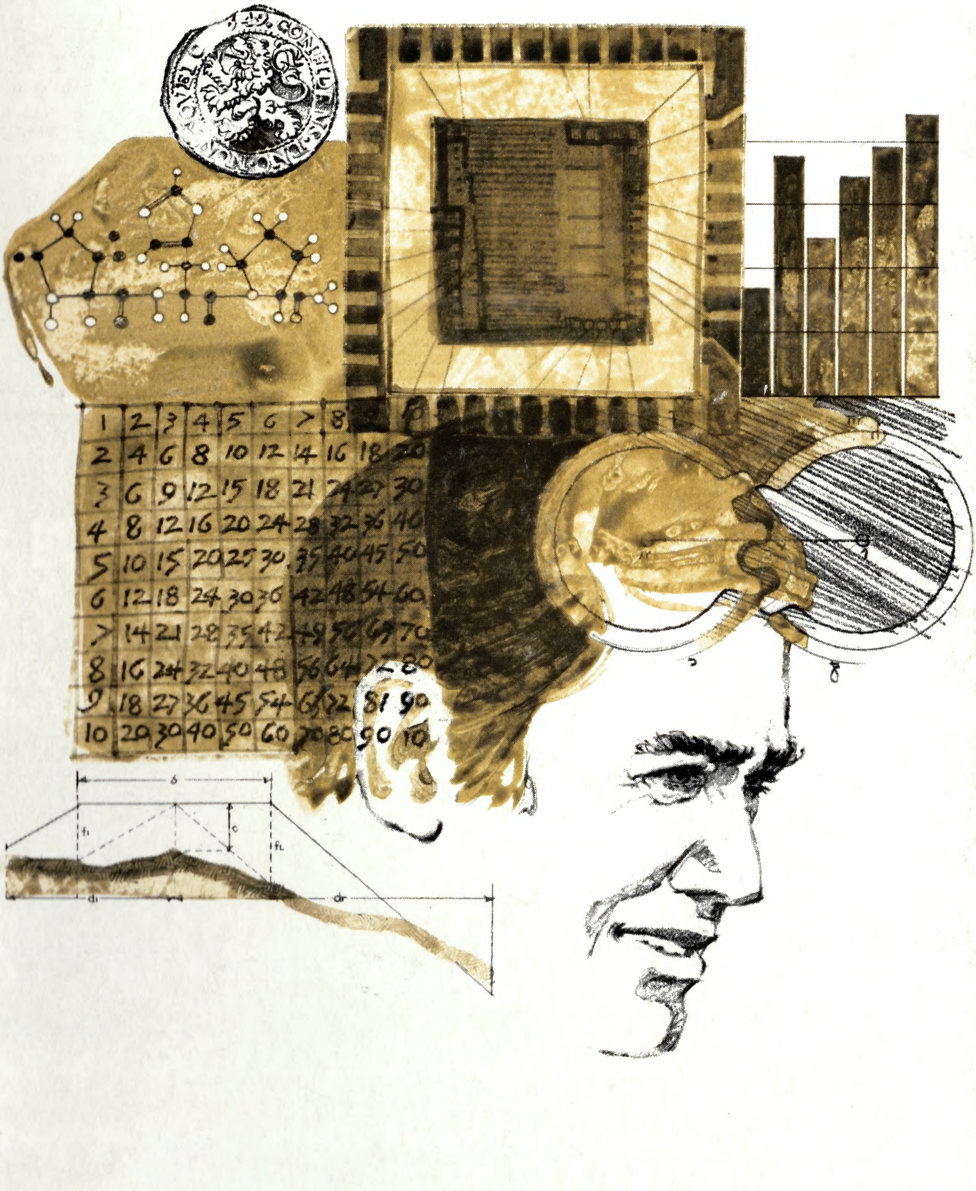
HEWLETT-PACKARD

HP-67

Standard Pac



The program material contained herein is supplied without representation or warranty of any kind. Hewlett-Packard Company therefore assumes no responsibility and shall have no liability, consequential or otherwise, of any kind arising from the use of this program material or any part thereof.

PRINTED IN SINGAPORE

Introduction

The HP-67 Standard Pac provides an excellent nucleus from which to build your program library. The programs address topics common to business, science, and engineering as well as providing enjoyable programs such as *Arithmetic Teacher, Follow Me, and Moon Rocket Lander.*

No knowledge of programming is required to use the programs in Standard Pac. However, familiarity with sections one through five of the Owner's Handbook (or previous HP calculator experience) is assumed. If this is your first encounter with programmability, be sure to read ''Running a Program'' on pages iv to xi of this manual. This detailed description is designed to help you become more familiar with your calculator. It is most effective when you perform all operations as they are described.

For each program the Standard Pac provides a description, user instructions, keystrokes for example problems, a prerecorded magnetic card (in the plastic card case) and program listings (at the back of this manual). There is also a diagnostic program for checking calculator operation, a head cleaning card which can be used occasionally to clean the magnetic card read/write head, and blank magnetic cards which may be used to record the programs you write.

Standard Pac differs from optional HP-67/97 application pacs in that it contains explanations of important programming techniques. The titles and page numbers of these explanations may be found opposite page 15-03 of this manual.

We hope you find Standard Pac useful in your daily calculations.

NOTES

CONTENTS

Program **Page**

1. Moving Average **01-01**
   Follows trends in data.
2. Tabulator **02-01**
   Adds columns and rows simultaneously for tabular data.
3. Curve Fitting **03-01**
   Fits straight lines, exponential curves, logarithmic curves or power curves to data.
4. Calendar Functions **04-01**
   Calculates days between dates, a future date or past date, or day of the week.
5. Annuities and Compound Amounts **05-01**
   Solves problems involving annuities or compound amounts.
6. Follow Me **06-01**
   The programmable program.
7. Triangle Solutions **07-01**
   Solves for the unknowns of any defined plane triangle.
8. Vector Operations **08-01**
   Addition, cross product, dot product, and coordinate transformation for two-dimensional and three-dimensional vectors.
9. Polynomial Evaluation **09-01**
   Solves cubic and quadratic equations and evaluates up to third degree polynomials for arbitrary real values of x.
10. Matrix Operations **10-01**
    Finds determinant and inverse for 3 × 3 system. Also, allows multiplication of 3 × 3 matrix by column matrix.
11. Calculus and Roots of f(x) **11-01**
    Approximates the derivative of a function at a point, evaluates a function at a point, and approximates the integral over a finite interval for a user specified function f(x). Also, approximates real roots of f(x).
12. English—SI Conversions (Metric Conversions) **12-01**
    Common unit conversions.
13. Arithmetic Teacher **13-01**
    Generates addition, subtraction, multiplication, and division problems for preschool and elementary students.
14. Moon Rocket Lander **14-01**
    Exciting action game simulating landing a rocket on the moon.
15. Diagnostic Program **15-01**
    Checks calculator functions.

Program Listings and Programming Techniques **L00-01**

RUNNING A PROGRAM

Loading A Program

Select the *Curve Fitting* card, SD-03A, from the card case supplied with this application pac.

Set W/PRGM-RUN switch to RUN.

Turn the calculator ON. You should see 0.00.

Gently insert either end of the card (printed side up) in the reader slot as shown in figure 1.

Figure 1.

When the card is part way in, a motor engages and passes it out the side of the calculator. Sometimes the motor engages but does not pull the card in. If this happens, push the card a little farther into the machine. Do not impede or force the card; let it move freely.

The display will show "Error" if the card reads improperly. In this case, press CL x and reinsert the card.

Since *Curve Fitting* is longer than 112 steps, the display now shows "Crd" indicating that a second card pass is necessary to load the remaining steps. With the writing still visible to you, insert the *opposite* end of the card (figure 2) and pass the card through the card reader again.

Figure 2.

When the motor stops, remove the card from the side of the calculator and insert it in the "window slot" of the calculator (see figure 3).

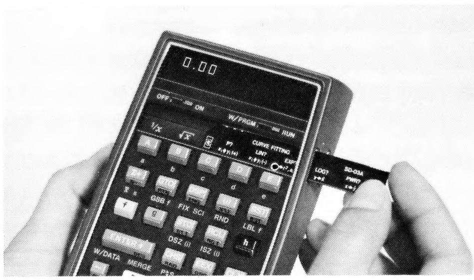


Figure 3.

The program has now been stored in the calculator. It will remain stored until another program is loaded or the calculator is turned off.

MAGNETIC CARD

Instructions On The Magnetic Card

Look at the card that you just inserted in the window slot of the calculator. The mnemonics on the card can help you run the program. The most important thing to note is that the mnemonics are associated with the user-definable keys A – E. For instance "LOG?" and "y →x̂" are associated with the D key.

Following is a table of the important types of symbols and conventions used in this pac. The table is provided as a reference until you become familiar with the symbols on the magnetic cards.

Symbols And Conventions

SYMBOL OR CONVENTION	INDICATED MEANING
White mnemonic: x A	White mnemonics are associated with the user-definable key they are above when the card is inserted in the calculator's window slot. In this case the value of x could be input by keying it in and pressing A.
Gold mnemonic: y x f E	Gold mnemonics are similar to white mnemonics except that the gold f key must be pressed before the user-definable key. In this case y could be input by pressing f E.
x ↑y A	↑ is the symbol for ENTER↑. In this case ENTER↑ is used to separate the input variables x and y. To input both x and y you would key in x, press ENTER↑, key in y and press A.
$\boxed{x}$ A	The box around the variable x indicates input by pressing STO A.
(x) A	Parentheses indicate an option. In this case, x is not a required input but could be input in special cases.
→x A	→ is the symbol for calculate. This indicates that you may calculate x by pressing key A.
→x, y, z A	This indicates that x, y, and z are calculated by pressing A once. The values would be sequentially displayed in x, y, z order.

SYMBOL OR CONVENTION	INDICATED MEANING
➜x; y; z A	The semi-colons indicate that after x has been calculated using A, y and z may be calculated in turn by pressing R/S and then again R/S.
➜"x ",y A	The quote marks indicate that the x value will be "paused" or held in the display for one second. The pause will be followed by the display of y.
◇ x A	The two-way arrow ◇ indicates that x may be either output or input when the associated user-definable key is pressed. If numeric keys have been pressed between user-definable keys, x is stored. If numeric keys have not been pressed, the program will calculate x.
P? A	The question mark indicates that this is a mode setting, while the mnemonic indicates the type of mode being set. In this case a pause mode is controlled. Mode settings typically have a 1.00 or 0.00 indicator displayed after they are executed. If 1.00 is displayed, the mode is on. If 0.00 is displayed, it is off.
START A	The word START is an example of a command. The start function should be performed to begin or start a program. It is included when initialization is necessary.
DEL A	This special command indicates that the last value or set of values input may be deleted by pressing A.

FORMAT OF USER INSTRUCTIONS

The completed User Instruction Form—which accompanies each program—is your guide to operating the programs in this Pac.

The form is composed of five labeled columns. Reading from left to right, the first column, labeled STEP, gives the instruction step number.

The INSTRUCTIONS column gives instructions and comments concerning the operations to be performed.

The INPUT-DATA/UNITS column specifies the input data, and the units of data if applicable. Data input keys consist of [0] to [9] and decimal point (the numeric keys), EEX (enter exponent), and CHS (change sign).

The KEYS column specifies the keys to be pressed after keying in the corresponding input data.

The OUTPUT-DATA/UNITS column specifies intermediate and final outputs and their units, where applicable.

The following illustrates the User Instruction Form for *Curve Fitting,* SD-03A.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	Load side 1 and side 2.			
2	Optional: Select pause input			
	mode.		f A	1.00/0.00
3	Select type of regression:			
	for linear fit		f B	1.00
	for exponential fit		f C	1.00
	for logarithmic fit		f D	1.00
	for power fit		f E	1.00
4	Input x value*.	x_i	ENTER↑	x_i
5	Input y value.	y_i	A	i + 1
6	Repeat steps 4 and 5 for all data			
	pairs**.			
7	Compute and output coefficient			
	of determination r^2 and a and b.		C	r^2, a, b
8	Optional: Make projections			
	based on a known y value.	y	D	$\hat{x}$

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
9	Optional: Make projections			
	based on a known x value.	x	E	ŷ
10	For a new case go to step 3.			
	*Note that this step may be skip-			
	ped if the x value equals the dis-			
	played counter (i + 1).			
	**The last set of data pairs may			
	be deleted by pressing h R↓			
	then B. Any set of data pairs			
	may be deleted by entering them			
	as in steps 4 and 5 and			
	pressing B.			

Since you loaded this program in "Loading A Program" on page iv, step 1 is already done and we can move to step 2. (If you turned your calculator off, you must reload the program.)

Step 2 is optional. It is primarily intended for printer control on the HP-97 printing programmable calculator. On your HP-67 calculator, print commands are interpreted as pause commands. That is, the calculator stops and displays the X-register value for one second and then continues with program execution.

In this particular application the print mode provides a permanent record of input data on the HP-97 printing calculator. On the HP-67 pocket calculator the input values are displayed for review if the print input mode is selected.

To select this "print/pause" mode, you would press f A as shown in the KEYS column of the User Instruction Form. Go ahead and press f A now. You should see a 1.00 in the display as indicated in the OUTPUT DATA/UNITS column. Successive presses of f A will cause 0.00 and 1.00 to be displayed alternately, indicating that the print/pause mode is off (0.00) or on (1.00). Try this, but leave 0.00 displayed (print/pause mode off) before moving to step 3.

In step 3 the type of curve fit is selected. There are four options listed, and you must select one. For example, to select exponential curve fit, refer to the *KEYS* column of the same line and press f C. Do this. The number 1.00 should be displayed, as shown in the OUTPUT-DATA/UNITS column.

The magnetic card gives short mnemonic hints about the four possible modes that may be selected. Printed in gold above the C key is "EXP?" indicating that the exponential mode is set by pressing f C.

To do a curve fit, you must input a number of data pairs (x_i and y_i). Steps 4, 5 and 6 give the input instructions. First key in x_i as indicated under INPUT-DATA/UNITS. Then press ENTER↑ to tell the calculator that you have completed building the number x. Then key in the value for y_i and press A. The number of data pairs plus one (i + 1) will appear in the display. Repeat the procedure for all data pairs. Try it for this data set:

x_i	1	3	7
y_i	2.7	20	1100

The keystrokes you should use are 1 ENTER↑ 2.7 A 3 ENTER↑ 20 A 7 ENTER↑ 1100 A. If you make a mistake, look at the second note at the bottom of the User Instructions. It describes procedures for correcting errors. If the last input pair was in error, you could press h R↓ B and eliminate it. Don't do this. Instead eliminate the (3,20) pair and replace it with (4,60). The keystrokes are 3 ENTER↑ 20 B 4 ENTER↑ 60 A.

Now that you know how the program works, the mnemonics on the magnetic card will prompt you on data input and data correction.

When all data have been keyed into the calculator, the regression coefficients can be calculated. Step 7 of the User Instructions says press C to do this.

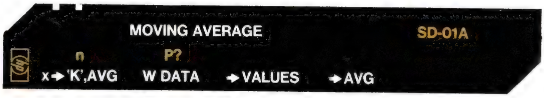
Three values will be displayed in the order listed in the comments column of the user instructions. First, the coefficient of determination (r^2 here equal to 1.00) will be displayed. Then the regression coefficients, a (1.02) and b (1.00), will be displayed. Go ahead and press C. When execution stops (after all three values have been displayed), you may review the values by pressing C again.

If you wish to have more time to observe a value during a pause, press R/S during the pause. This stops program execution leaving the value displayed. To restart the calculator, press R/S again. Try this. Press C, then stop the calculator during the first pause by pressing R/S. Press R/S again to restart program execution. Stop the calculator during the second pause and see 1.02. Press R/S again to complete the calculation. Note that during an output pause, the decimal point flashes. This signifies that program execution has not terminated and will resume automatically.

Now try a projection. Step 9 instructs you to key in an x value, press **E** and see a projected $\hat{y}$ value. Try an x value of 10. You should see a projected $\hat{y}$ result of 22926.17. You can also estimate an x value $\hat{x}$ using a known y value. Leave the value of 22926.17 in the display and press **D**. The value 10.00 should be displayed again.

If your answers agree with ours, you are ready to try other programs in Standard Pac. If your answers did not agree with ours, try the procedure again.

MOVING AVERAGE



In a moving average, a specified number of data points are averaged. When there is a new piece of input data, the oldest piece of data is discarded to make room for the latest input. This replacement scheme makes the moving average a valuable tool in following trends. The fewer the number of data points, the more trend sensitive the average becomes. With a large number of data points, the average behaves more like a regular average, responding slowly to new input data.

This program allows for a moving average span of 1 to 22 units. The number of units, n, must be specified before any data input begins by keying it in and pressing f A. Then the data is input by keying in each value, x_k, and pressing A in turn. The calculator will display the current input number, k, until at least n values have been entered. After the n^{th} value (and for all succeeding values), the calculator will flash the current input number before halting with the moving average, AVG, in the display.

In many applications moving averages are calculated daily, weekly, monthly, or even yearly. In such cases it is necessary to store the register contents on a magnetic card for future use. To do this, press B for WRITE DATA and insert one side of the blank card. If the display says ''Crd'' after the first card pass, insert the other end of the card. If the display is unchanged after the first pass, all data has been recorded on the first pass and you may proceed to other calculations. When the recorded data is required again, insert the data card. If ''Crd'' appears after the first pass, load the other end of the card. The original data has been returned to the storage registers and you are ready to continue the moving average at the point you left off.

The value of the average may be displayed at any time by pressing D. This feature allows the average to be calculated before n data points have been input. The average is based on the number of inputs or n, whichever is smaller.

Remarks:

Attempts to input a value larger than 22.00 or smaller than 1.00 for n will result in a flashing display which can be cleared by pressing R/S.

All data storage registers are used.

Moving averages of 10.00 or more units require two passes of the data card to record or store the values.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	Load side 1 and side 2.			
2	If data from a previous calcu-			
	lation is stored on a magnetic			
	card, insert the magnetic			
	card and skip to step 5.			
3	Input number of points in			
	average (1 ≤ n ≤ 22)	n	f A	n
4	Optional: Select pause input		f B	1.00/0.00
	mode.			
5	Input data point and compute			
	moving average.*	x_k	A	"k", AVG
6	Go to step 5 for next input.			
7	Optional: To store data on			
	magnetic card for future use,			
	press B and insert card in			
	reader.		B	Crd
8	Optional: Output values in			
	newest to oldest order.		C	Values
9	Optional: Display average at			
	any time.		D	AVG
	For a new case go to step 2.			
	*If you make an error on data			
	input, you must start over unless			
	you previously recorded data			
	on a magnetic card. If data was			
	previously recorded, load the			
	data card and start with the first			
	value input after recording the			
	card.			

Example 1:

A six-period moving average is used to project monthly sales. The first 6 months of sales are as follows:

Month	1	2	3	4	5	6
Sales	125	183	207	222	198	240

Compute the moving average. Also compute the average after month three.

Keystrokes:	Outputs:	
6 f A	6.00	
125 A	1.00	
183 A	2.00	
207 A	3.00	
D	171.67	(average after month three)
222 A	4.00	
198 A	5.00	
240 A	"6.00",	195.83

Now record the data for example 2.

B ⟶ Crd

Insert a blank magnetic card in the card reader.

Now turn the calculator off and assume a month has passed. Turn the calculator back on and load both sides of *Moving Average*.

Example 2:

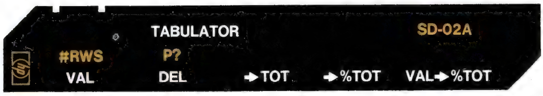
The actual sales for the seventh month totaled 225 units. Compute a new moving average with this data. Also, output the values in the average.

Load the magnetic data card recorded at the end of example 1.

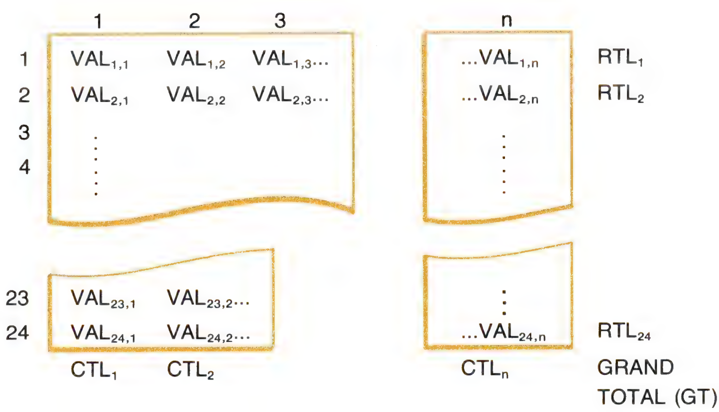
Keystrokes:	Outputs:	
225 A	"7.00",	212.50
C	225.00 ***	(current moving
	240.00 ***	average values
	198.00 ***	in newest to
	222.00 ***	oldest order)
	207.00 ***	
	183.00 ***	
	6.00	

NOTES

TABULATOR



This program is designed to be of aid in tabulating applications such as accounting and estimating. It can be used to add single columns containing up to 24 values (VAL), remember each value, and find the percent of total of each value. (The first example problem shows this type of use.) The program can also be used to total any number of columns and find row totals, the percent of total for each row total, and the grand total for a table of values. The total of each column is displayed as soon as the column is completed.



Column totals (CTL) are output when the column is complete.

Figure 1

Equations:

$$\% \text{ of Total}_i = \frac{\text{Row Total}_i}{\text{Grand Total}} \times 100$$

Remarks:

If the last value input was in error, it may be deleted by pressing **B**. This subtracts the value from both column and row totals and resets the indices.

Attempts to specify more than 24 or less than 1 for the number of rows will result in flashing input which can be cleared by pressing R/S.

All data storage registers are used.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	Load side 1 and side 2.			
2	Key in number of rows (1 to 24)			
	and initialize*.	ROWS	f A	0.00
3	Optional: Select pause input			
	mode		f B	1.00/0.00
4	Input value	VAL	A	VAL (or CTL)
5	If your last data input was in			
	error execute this step to return			
	to prior status:		B	
6	Go to step 4 until all values have			
	been input.			
7	Obtain outputs:			
	Output row totals and grand total.		C	ROWS
	or			
	Output % of grand total for each			
	row total.		D	ROW %
8	Optional: Compute percentage			
	of grand total for any number.	NUMBER	E	% of GT
9	For new case go to step 2.			
	*Flashing input indicates an			
	input less than one or greater			
	than 24. Clear with R/S.			

Example 1:

The following list of unit sales figures are to be totaled and converted to monthly percentages.

January: 1012	May: 1502	September: 1051
February: 1235	June: 1073	October: 1244
March: 895	July: 973	November: 1127
April: 1123	August: 1250	December: 977

Keystrokes:	Output:	
12 [I] [A] →	0.00	
1012 [A] 1235 [A] 895 [A] 1123 [A] →	1123.00	
1502 [A] 1073 [A] 973 [A] 1250 [A] →	1250.00	
1051 [A] 1244 [A] 1127 [A] 977 [A] →	13462.00	
[D] →	7.52 ***	(Percents)
	9.17 ***	
	6.65 ***	
	8.34 ***	
	11.16 ***	
	7.97 ***	
	7.23 ***	
	9.29 ***	
	7.81 ***	
	9.24 ***	
	8.37 ***	
	7.26 ***	
	100.00 ***	
[C] →	1012.00 ***	(row totals)
	1235.00 ***	
	895.00 ***	
	1123.00 ***	
	1502.00 ***	
	1073.00 ***	
	973.00 ***	
	1250.00 ***	
	1051.00 ***	
	1244.00 ***	
	1127.00 ***	
	977.00 ***	
	13462.00 ***	

Example 2:

The following table is to be totaled (both rows and columns). Also, find the percent of total sales for each booklet.

BOOKLET SALES DATA

	JAN	FEB	MARCH	APRIL	MAY
BOOK 1	273	284	303	244	252
BOOK 2	1093	847	1222	1027	978
BOOK 3	423	654	683	540	570
BOOK 4	118	255	453	755	805

Keystrokes:	**Outputs:**	
4 f A	0.00	
273 A 1093 A 423 A 118 A	1907.00	(Jan total)
284 A 847 A 654 A 255 A	2040.00	(Feb total)
303 A 1222 A 683 A 453 A	2661.00	(Mar total)
244 A 1027 A 540 A 755 A	2566.00	(Apr total)
252 A 978 A 570 A 805 A	2605.00	(May total)
C	Row totals	
D	% of row totals	

BOOKLET SALES DATA

	JAN	FEB	MARCH	APRIL	MAY	TOTALS	PERCENTS
BOOK 1	273	284	303	244	252	1356	11.51%
BOOK 2	1093	847	1222	1027	978	5167	43.87%
BOOK 3	423	654	683	540	570	2870	24.37%
BOOK 4	118	255	453	755	805	2386	20.26%
TOTALS	1907	2040	2661	2566	2605	11779.00	100.00%

CURVE FITTING

This program can be used to fit data to:

1. Straight lines (linear regression); $y = a + bx$,
2. Exponential curves; $y = ae^{bx}$ $(a > 0)$,
3. Logarithmic curves; $y = a + b \ln x$,
4. Power curves; $y = ax^b$ $(a > 0)$.

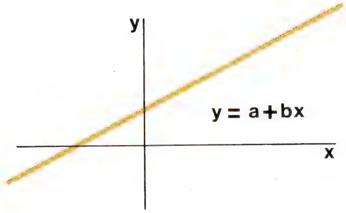
The type of curve fit must be determined before data input begins. To select linear regression, you would press the **f** **B** keys. To select exponential curve fit, press **f** **C**. To select logarithmic curve fit, press **f** **D**. To select power curve fit, press **f** **E**. Do not attempt to change from one type of fit to another after data input has begun because the summation registers are cleared when the type of curve fit is selected. Restarting can be accomplished by repeating the curve fit selection process.

Data pairs (x_i and y_i) are input by keying in x_i, pressing **ENTER↑**, keying in y_i and pressing the **A** key. Any number of data pairs may be input. If, after pressing the **A** key, you discover a data pair was incorrect, wait until execution stops, press **h** **R↓**, then the **B** key. This will eliminate the errant data pair. If you wish to eliminate any data pair previously input, key it in (x **ENTER↑** y) and press **B**.

After all data pairs have been input, press **C**. This initiates calculation and output of the coefficient of determination r^2, and the regression coefficients a and b. The coefficient of determination indicates the quality of fit achieved by the regression. Values of r^2 close to 1.00 indicate a better fit than values close to zero. The regression coefficients a and b define the curve generated, according to the equations at the beginning of this discussion.

After the regression coefficients have been calculated, projections may be made based on the curve fit. Key in a known x value, press **E** and see an estimated y value, $\hat{y}$, or key in a known y value, press **D** and see an estimated x value, $\hat{x}$.

Linear Regression

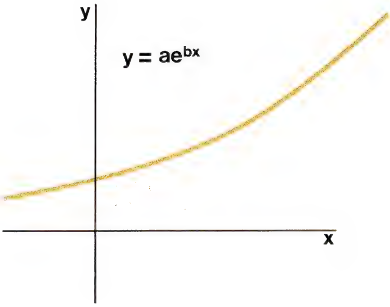


$$b = \frac{\Sigma x_i y_i - \dfrac{\Sigma x_i \ \Sigma y_i}{n}}{\Sigma x_i^2 - \dfrac{(\Sigma x_i)^2}{n}}$$

$$a = \left[\frac{\Sigma y_i}{n} - b\,\frac{\Sigma x_i}{n}\right]$$

$$r^2 = \frac{\left[\Sigma x_i y_i - \dfrac{\Sigma x_i \ \Sigma y_i}{n}\right]^2}{\left[\Sigma x_i^2 - \dfrac{(\Sigma x_i)^2}{n}\right]\left[\Sigma y_i^2 - \dfrac{(\Sigma y_i)^2}{n}\right]}$$

Exponential Curve Fit

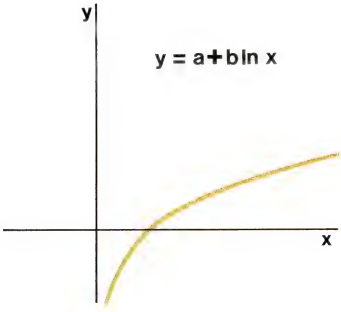


$$b = \frac{\Sigma x_i \ln y_i - \dfrac{1}{n}(\Sigma x_i)(\Sigma \ln y_i)}{\Sigma x_i^2 - \dfrac{1}{n}(\Sigma x_i)^2}$$

$$a = \exp\left[\frac{\Sigma \ln y_i}{n} - b\frac{\Sigma x_i}{n}\right]$$

$$r^2 = \frac{\left[\Sigma x_i \ln y_i - \frac{1}{n}\Sigma x_i \Sigma \ln y_i\right]^2}{\left[\Sigma x_i^2 - \frac{(\Sigma x_i)^2}{n}\right]\left[\Sigma(\ln y_i)^2 - \frac{(\Sigma \ln y_i)^2}{n}\right]}$$

Logarithmic Curve Fit

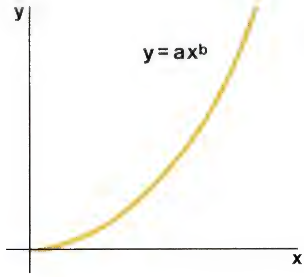


$$b = \frac{\Sigma y_i \ln x_i - \frac{1}{n}\Sigma \ln x_i \Sigma y_i}{\Sigma(\ln x_i)^2 - \frac{1}{n}(\Sigma \ln x_i)^2}$$

$$a = \frac{1}{n}(\Sigma y_i - b\,\Sigma \ln x_i)$$

$$r^2 = \frac{\left[\Sigma y_i \ln x_i - \frac{1}{n}\Sigma \ln x_i \Sigma y_i\right]^2}{\left[\Sigma(\ln x_i)^2 - \frac{1}{n}(\Sigma \ln x_i)^2\right]\left[\Sigma y_i^2 - \frac{1}{n}(\Sigma y_i)^2\right]}$$

Power Curve Fit



$$b = \frac{\Sigma(\ln x_i)(\ln y_i) - \dfrac{(\Sigma \ln x_i)(\Sigma \ln y_i)}{n}}{\Sigma(\ln x_i)^2 - \dfrac{(\Sigma \ln x_i)^2}{n}}$$

$$a = \exp\left[\frac{\Sigma \ln y_i}{n} - b\,\frac{\Sigma \ln x_i}{n}\right]$$

$$r^2 = \frac{\left[\Sigma(\ln x_i)(\ln y_i) - \dfrac{(\Sigma \ln x_i)(\Sigma \ln y_i)}{n}\right]^2}{\left[\Sigma(\ln x_i)^2 - \dfrac{(\Sigma \ln x_i)^2}{n}\right]\left[\Sigma(\ln y_i)^2 - \dfrac{(\Sigma \ln y_i)^2}{n}\right]}$$

Remarks:

Negative and zero values of x_i will cause a machine error for logarithmic curve fits. Negative and zero values of y_i will cause a machine error for exponential curve fits. For power curve fits both x_i and y_i must be positive, non-zero values.

Registers R_0–R_9 are available for user storage.

It is not necessary to key in the x value if it corresponds to the counter returned to the display (see example 1).

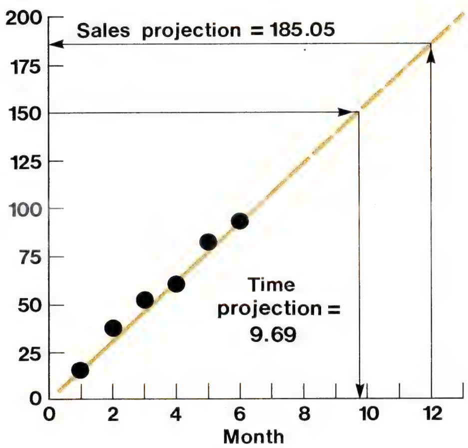
As the differences between x and/or y values become small, the accuracy of the regression coefficients will decrease.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	Load side 1 and side 2.			
2	Optional: Select pause input			
	mode.		f A	1.00/0.00
3	Select type of regression:			
	for linear fit		f B	1.00
	for exponential fit		f C	1.00
	for logarithmic fit		f D	1.00
	for power fit		f E	1.00
4	Input x value*.	x_i	ENTER↑	x_i
5	Input y value.	y_i	A	i + 1
6	Repeat steps 4 and 5 for all data			
	pairs**.			
7	Compute and output coefficient			
	of determination r^2 and a and b.		C	r^2, a, b
8	Optional: Make projections			
	based on a known y value.	y	D	$\hat{x}$
9	Optional: Make projections			
	based on a known x value.	x	E	$\hat{y}$
10	For a new case go to step 3.			
	*Note that this step may be skip-			
	ped if the x value equals the dis-			
	played counter (i + 1).			
	**The last set of data pairs may			
	be deleted by pressing h R↓			
	then B. Any set of data pairs may			
	be deleted by entering them as in			
	steps 4 and 5 and pressing B.			

Example 1:

Below is the sales data for the first 6 months of a product's life. According to a linear projection, what should the sales be after 12 months? When would sales reach the 150 unit per month mark assuming constant linear growth.

Month	1	2	3	4	5	6
Sales	15	37	52	59	83	92



Keystrokes:	**Outputs:**	
f B →	1.00	
15 A 37 A 52 A 59 A 83 A 92 A →	7.00	
C →	0.98 ***	(r^2)
	3.33 ***	(a)
	15.14 ***	(b)
12 E →	185.05	units
150 D →	9.69	months

Example 2:

The velocity of a particle experiencing constant acceleration is expressed by

$$v = v_o + \alpha t$$

where v is the velocity, v_0 is the initial velocity, α is the acceleration and t is the time since $v = v_0$.

The following time velocity data was experimentally obtained for a particle:

t (sec)	V(m/sec)
5	140
6	149
7	159
9	175

What was the velocity at t = 0? What will the velocity be when t = 20?

Note that the equation for velocity

$$v = v_o + \alpha t$$

is the equation of a straight line and is analogous to

$$y = a + bx$$

Therefore use linear regression with v substituted for y, v_o for a, α (acceleration) for b and t for x.

Keystrokes:	**Outputs:**	
f B →	1.00	
5 ENTER↑ 140 A 6 ENTER↑ 149 A		
7 ENTER↑ 159 A →	4.00	
9 ENTER↑ 175 A C →	1.00 ***	(r^2)
	96.54 ***	(a, v_o)
	8.77 ***	(b, acceleration)
20 E →	271.97	(m/sec)

Example 3:

Many compression processes can be correlated using the power curve

$$p = av^{-b}$$

where b is the polytropic constant of the process.

Pressure-volume data for a compression process is shown below. Run a power curve fit to determine the polytropic constant, −b. What is the pressure when v is 15?

v	p
10	210
30	40
50	12
70	9
90	6.8

Keystrokes:	**Outputs:**
f E	1.00
10 ENTER↑ 210 A 30 ENTER↑ 40 A	
50 ENTER↑ 12 A	4.00
70 ENTER↑ 9 A 90 ENTER↑ 6.8 A C	0.99 *** (r^2)
	8599.81 *** (a)
	–1.62 *** (–b)
15 E	108.35

CALENDAR FUNCTIONS

For the period March 1, 1900 through February 28, 2100, this program interchangeably solves for dates and days. Given two dates, the number of days between them can be calculated. Given one date and a specified number of days, a second date can be found. The program will also work in terms of weeks between dates or compute the day of the week given the date. After input of a date, its Julian Day number* is displayed.

A date must be input in mm.ddyyyy format. For instance, June 3, 1975 is keyed in as 6.031975. It is important that the zero between the decimal point and the day of the month be included when the day of the month is less than 10. Weeks are input and output as WKS.DYS. Seven weeks, three days would be 7.3. The day of the week is represented by the digits 0 through 6 where zero is Sunday.

Equations:

To compute the day number from the date:

Julian Day number = INT (365.25 y′) + INT (30.6001 m′) + d + 1,720,982

where

$$y' = \begin{cases} \text{year} - 1 & \text{if m = 1 or 2} \\ \text{year} & \text{if m > 2} \end{cases}$$

$$m' = \begin{cases} \text{month} + 13 & \text{if m = 1 or 2} \\ \text{month} + 1 & \text{if m > 2} \end{cases}$$

Then days between dates is found by

$$\text{Days} = \text{Day number}_2 - \text{Day number}_1$$

To compute the date from a day number:

$$\text{Day \#} = \text{Julian Day Number} - 1{,}720{,}982$$

$$y' = \text{INT}\left[\frac{\text{Day \#} - 122.1}{365.25}\right]$$

*The Julian Day number is an astronomical convention representing the number of days since January 1, 4713 B.C.

$$m' = \text{INT}\left[\frac{\text{Day \#} - \text{INT}(365.25\ y')}{30.6001}\right]$$

$$\text{Day of the month} = \text{Day \#} - \text{INT}\left[365.25\ y'\right] - \text{INT}\left[30.6001\ m'\right]$$

$$\text{Month} = m = \begin{cases} m' - 13 \text{ if } m' = 14 \text{ or } 15 \\ m' - 1 \text{ if } m' < 14 \end{cases}$$

$$\text{Year} = \begin{cases} y' & \text{if } m > 2 \\ y' + 1 & \text{if } m = 1 \text{ or } 2 \end{cases}$$

To compute the day of the week:

$$\text{Day of the week} = 7 \times \text{FRAC}\left[(\text{Day \#} + 5)/7\right]$$

Remarks:

No checking is done to determine if input data represents valid dates.

In this program the calculator uses flag 3 to decide what to do after **A**, **B**, **C** or **D** is pressed. If the numeric keys have been pressed, flag 3 is on. This causes the value in the display to be stored as an input when the user-definable key is pressed. If no numeric keys have been touched, the program will calculate the value associated with the user-definable key. Thus, it is important not to touch the numeric keys between the last input and the attempt to calculate a result.

Registers R_0–R_2, R_B, R_D, R_E and R_{S0}–R_{S9} are available for user storage.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	Load side 1 and side 2.			
2	For day of the week calculations			
	go to step 6.			
3	Input two of the following:			
	First date (mm.ddyyyy)	DT_1	A	Day $\#_1$
	Second date (mm.ddyyyy)	DT_2	B	Day $\#_2$
	Days between dates	DAYS	C	Days
	or weeks between dates*	WKS. DYS	D	Days

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
4	Calculate one of the following:			
	First date		A	DT_1
	Second date		B	DT_2
	Days between dates		C	Days
	Weeks between dates		D	WKS. DYS
5	For a new case go to step 2.			
6	Input date and calculate day			
	of the week (0 = Sunday,			
	6 = Saturday).	DT	E	DOW
7	For a new case go to step 2.			
	*Either days between dates or			
	weeks between dates, but not			
	both, may be input in step 3.			

Example 1:

Senior Lieutenant Yuri Gagarin flew Vostok I into space on April 12, 1961. On July 21, 1969 Neil Armstrong set foot on the moon. How many days had passed between the first manned space flight and the moon landing? How many weeks and days? On what day of the week did each event take place?

Keystrokes:	Outputs:	
4.121961 A 7.211969 B C →	3022.	(days)
D →	431.5	(weeks.days)
4.121961 E →	3.	(Wednesday)
7.211969 E →	1.	(Monday)

Example 2;

A short term note is due in 200 days. If the issue date is June 11, 1976, what is the maturity date?*

Keystrokes:	**Outputs:**	
6.111976 A 200 C B ⟶	12.281976	(December 28, 1976)

*Some securities use a 30/360 day calendar while this program performs all calculations using the actual number of days. Do not use the program for financial purposes unless you are sure that actual calendar days are correct.

ANNUITIES AND COMPOUND AMOUNTS

This program can be used to solve a variety of problems involving money, time and interest. The following variables can be inputs or outputs:

- n, which is the number of compounding periods. (For a 30 year loan with monthly payments, n = 12 × 30 = 360.)
- i, which is the periodic interest rate expressed as a percent. (For other than annual compounding, divide the annual percentage rate by the number of compounding periods in a year; i.e. 8% annual interest compounded monthly equals 8/12 or 0.667%.)
- PMT, which is the periodic payment.
- PV, which is the present value of the cash flows or compound amounts.
- FV, which is the future value of a compounded amount or a series of cash flows.
- BAL, which is the balloon or remaining balance at the end of a series of payments.

The program accommodates payments which are made at the end of compounding periods or at the beginning. Payments made at the end of compounding periods (ordinary annuity) are common in direct reduction loans and mortgages while payments at the beginning of compounding periods (annuity due) are common in leasing. When the program is loaded into the calculator or when the START function **f** **A** is executed, the calculator is set in ordinary annuity mode. Pressing **f** **B** sets the calculator in annuity due mode and displays 1.00 indicating that the annuity due mode is set. Pressing **f** **B** again returns the machine to ordinary annuity mode and displays 0.00. Successive use of **f** **B** will alternately display 1.00 and 0.00 indicating that the annuity due mode is on or off, respectively.

In this program **STO** **A** is used to input n, **STO** **B** to input i, **STO** **C** to input PMT, **STO** **D** to input PV and **STO** **E** to input FV or BAL. After all inputs are stored it is possible to calculate the unknown value by pressing the appropriate user-definable key. For instance, you would press **B** to calculate interest.

The START function (**f** **A**) performs two functions:

1. It sets PMT, PV, and BAL to zero (n and i are not affected).
2. It sets the ordinary annuity mode.

START provides a safe, convenient, easy to remember method of preparing the calculator for a new problem. It is not necessary to use START between problems containing the same combination of variables. For instance, any number of n, i, PMT, FV problems involving different numbers and/or different combinations of knowns could be done in succession without using START. Only the values which change from problem to problem would have to be keyed in. To change the combination of variables without using START, simply input zero for any variable which is no longer applicable. To go from n, i, PMT, PV problems to n, i, PV, FV problems, a zero would be stored (0 STO C) in place of PMT. Table 1 summarizes these procedures. START should always be used immediately after loading *Annuities and Compound Amounts*.

Table I

Possible Solutions Using *Annuities and Compound Amounts*

Allowable Combination of Variables	Applications		Initial Procedure
	Ordinary Annuity	Annuity Due	
n, i, PMT, PV (Input any three and calculate the fourth.)	Direct reduction loan Discounted notes Mortgages	Leases	Use START or set BAL to zero.
n, i, PMT, PV, BAL (Input any four and calculate the fifth.)	Direct reduction loan with balloon Discounted notes with balloon	Leases with residual values	None
n, i, PMT, FV (Input any three and calculate the fourth.)	Sinking fund	Periodic savings insurance	Use START or set PV to zero.
n, i, PV, FV (Input any three and calculate the fourth.)	Compound amount Savings (Annuity mode is not applicable and has no effect)		Use START or set PMT to zero.

Equations:

$$PV = \pm \frac{PMT}{i} A \left[1 - (1+i)^{-n}\right] + (\text{BAL or FV})(1+i)^{-n}$$

where

$$A = \begin{cases} 1 & \text{ordinary annuity} \\ (1+i) & \text{annuity due.} \end{cases}$$

The sign is plus if FV is zero and minus if PV is zero.

Remarks:

The calculator must be in FIX display mode to solve for i when payments are involved.

The equation above is solved for i using Newton's method where:

$$i_n = i_{n-1} - \frac{f\,(i_{n-1})}{f'(i_{n-1})}$$

This is why solutions involving PMT and i take longer than other solutions. The algorithm works best for positive input values and for interest rates between zero and 100%. It is quite possible to define problems which cannot be solved by this technique. Such problems usually result in an error message but may simply continue to run indefinitely.

Iterative interest solutions are accurate to the number of significant figures of the display setting. It is possible to obtain more significant figures by changing the display setting from DSP 2 to DSP 3, DSP 4, DSP 5, etc. However, time for solution increases as accuracy is improved.

Problems with negative balloon payments may have more than one mathematically correct answer (or no answer at all). While this program may find one of the answers, it has no way of finding or indicating other possibilities.

RCL A, RCL B, RCL C, RCL D and RCL E may be used to review associated values at any time.

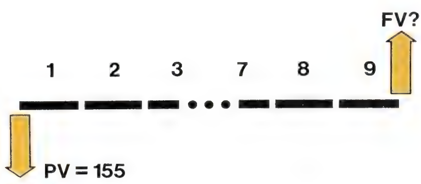
Registers R_0–R_2 and R_{S0}–R_{S9} are available for user storage.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	Load side 1 and side 2.			
2	Initialize		f A	0.00
3	If payments occur at the begin-			
	ning of the period set annuity			
	due mode*.		f B	1.00/0.00
4	Input the known values:			
	Number of periods	n	STO A	n
	Periodic interest rate	i (%)	STO B	i (%)
	Periodic payment	PMT	STO C	PMT
	Present value	PV	STO D	PV
	Future value, balloon or balance	FV, (BAL)	STO E	FV, (BAL)

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
5	Calculate the unknown value.			
	Number of periods		A	n
	Periodic interest rate		B	i (%)
	Periodic payment		C	PMT
	Present value		D	PV
	Future value, balloon or balance		E	FV, (BAL)
6	Output values in n, i, PMT, PV,			
	FV-BAL order.		f C	Values
7	For a new case, go to step 4			
	and change appropriate values.			
	Input zero for any value not			
	applicable in the new case.			
	*One or zero will be displayed			
	alternately after pressing f B,			
	indicating that the annuity			
	due mode is on or off.			

Example 1:

If $155 is placed in a savings account paying 5¾% compounded monthly, what sum of money will be in the account at the end of 9 years?



Keystrokes: | **Outputs:**

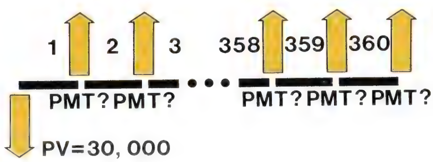
Keystrokes	Outputs
f A 155 STO D	155.00
5.75 ENTER↑ 12 ÷ STO B	0.48
9 ENTER↑ 12 × STO A	108.00
E	259.74

If the interest is changed to 6%, what is the sum?

6 ENTER↑ 12 ÷ STO B ——→ 0.50
E ——→ 265.62

Example 2:

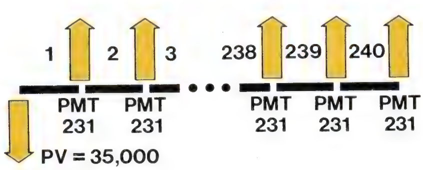
What is the monthly payment required to fully amortize a 30 year, $30,000 mortgage if the annual percentage rate is 9%? After solving the problem, review the values.



Keystrokes:	**Outputs:**	
f A 30 ENTER↑ 12 × STO A ——→	360.00	
30000 STO D ——→	30000.00	
9 ENTER↑ 12 ÷ STO B ——→	0.75	
C ——→	241.39	
f C ——→	360.00 ***	(n)
	0.75 ***	(i)
	241.39 ***	(PMT)
	30000.00 ***	(PV)
	0.00 ***	(FV)

Example 3:

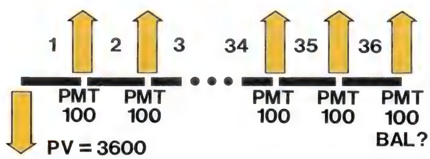
A fixed term annuity is available which requires a $35,000 initial deposit. In return the depositor will receive monthly payments of $231 for 20 years. What annual interest rate is being applied?



Keystrokes:	Outputs:	
f A 35000 STO D	35000.00	
231 STO C	231.00	
20 ENTER↑ 12 × STO A	240.00	
B	0.42	(0.42% monthly)
12 ×	5.00	(5% annual interest rate)

Example 4:

Two individuals are constructing a loan with a balloon payment. The loan amount is $3,600 and it is agreed that the annual interest rate will be 10% with 36 monthly payments of $100. What balloon payment amount, to be paid coincident with the 36th payment, is required to fulfill the loan agreement?

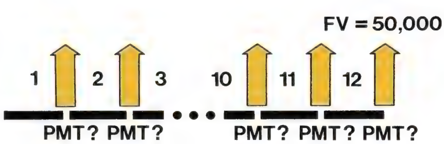


Keystrokes:	Outputs:
f A 3600 STO D 10 ENTER↑ 12 ÷	
STO B 36 STO A 100 STO C E	675.27

(Note that the final payment is $675.27 + $100.00 = $775.27 since the final payment falls at the end of the last period.)

Example 5:

A corporation has determined that a certain piece of equipment costing $50,000 will be required in 3 years. Assuming a fund paying 7% compounded quarterly is available, what quarterly payment must be placed in the fund in order to cover this cost if savings are to start at the end of this quarter?



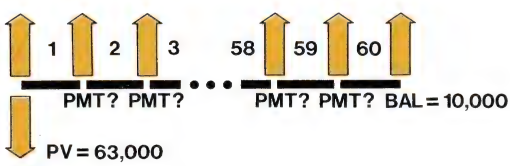
Keystrokes:	Outputs:
f A 50000 STO E 3 ENTER↑ 4 ×	
STO A 7 ENTER↑ 4 ÷ STO B C →	3780.69

What single amount, invested immediately, would provide the same effect?

0 STO C D →	40602.89

Example 6:

A "third party" leasing firm is considering the purchase of a mini-computer priced at $63,000 and intends to achieve a 13% annual yield by leasing the computer to a customer for a 5-year period. Ownership is retained by the leasing firm, and at the end of the lease they expect to be able to sell the equipment for at least $10,000. What should they establish as the monthly payments in order to realize their desired yield? (Since lease payments occur at the start of the periods, this is an annuity due problem.)



Keystrokes:	Outputs:
f A f B 63000 STO D 13 ENTER↑ 12 ÷	
STO B 5 ENTER↑ 12 × STO A	
10000 STO E C →	1300.16

If the price increased to $70,000, what should the payments be?

70000 STO D C →	1457.73

If the payments were increased to $1500 what would the yield be?

1500 STO C B →	1.18	(% per month)
12 × →	14.12	(% per year)

For more accuracy in calculation of the interest rate, change the display setting to five places and calculate the interest rate.

DSP 5 B ⟶ 1.17700
12 × ⟶ 14.12399

Return display to two places.

DSP 2 ⟶ 14.12

FOLLOW ME

This program allows the calculator to learn a simple set of keystrokes and repeat them over and over with different data. The allowable functions are plus, minus, times, divide, percent, constant and input-output halt. Up to 23 operations may be included in a sequence. Constants count as two operations each.

To run the program you would press **A** to start. Then do the first of the desired calculations using the +, −, ×, ÷, and % functions on the card. Any constants that repeat between problems should be followed by the **C** key so they will be automatically introduced at the proper times. Where intermediate answers or inputs are required, press **B** for an I/O halt. To signify the end of the sequence press **D**.

After the sequence has been learned by the calculator, only variables need be keyed in at I/O halts. The **E** key is used to start execution after I/O halts.

If an error is made while running a sequence, press **D** to start over. If an error is made while teaching the calculator a sequence, press **A** for a restart.

FOLLOW ME INSTRUCTION SET

Program Control	Action
START	Clears program from *Follow Me* memory and prepares for a new program sequence.
END	Defines the end of a sequence of keystrokes and resets program counter to the beginning of *Follow Me* memory.
FOLLOW	Starts halted program.
Programmable Operations	
+	Adds content of X register and Y register leaving result in X register.
−	Subtracts content of X register from Y register leaving result in X register.

Program Control	Action
×	Multiplies content of X register by content of Y register leaving result in X register.
÷	Divides content of Y register by content of X register leaving result in X register.
%	Multiplies content of Y register by content of X register divided by 100, replaces X register content with result and leaves content of Y register undisturbed.
CNST	Recalls constant to X register (requires two steps).
I/O	Input or output halt causes *Follow Me* to stop for display of calculated results and/or input of variables.

Remarks:

All four registers of the operational stack are available for input and output of data. By using all four registers the need for I/O halts can be minimized. Keyboard functions other than +, −, ×, ÷ and % may be used during I/O halts, but cannot be incorporated in a *Follow Me* program.

All data storage registers are used.

A flashing 24 results if more than 23 operations are attempted. This error condition may be cleared by pressing **R/S**.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	Load side 1 and side 2.			
2	Initialize.		**A**	0.00
3	Perform first string calculation			
	by pressing **B** at each point			
	where a halt for input or output			
	is desired, **C** after each con-			
	stant, **f** **A** for each addition,			
	f **B** for each subtraction,			
	f **C** for each multiplication, **f**			
	D for each division and **f** **E**			
	for percent operations. 23			
	steps are allowed (constants			
	count as two steps).			

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
4	End calculation		D	0.00
5	Key in variable(s) and initiate			
	execution	VAR	E	OUTPUT
6	If an error was made in step 5			
	go to step 4 and restart.			
7	Go to step five until calculation			
	is complete.			
8	For a new calculation of the			
	same type, go to step 5.			
9	For a new type of calculation,			
	go to step 2.			

Example 1:

Using *Follow Me,* program

$$y = 3(P + Q)$$

and calculate y for the following data:

P	Q
6	4
5	8
9	11

A solution:

Keystrokes: **Outputs:**

(Start)

A ——→ 0.00

(I/O) (I/O) (+) (×)

3 B 6 B 4 f A f C ——→ 30.00

(End)

D ——→ 0.00

3 E 5 E 8 E ——→ 39.00

3 E 9 E 11 E ——→ 60.00

A better solution:

Keystrokes:	Outputs:
[A]	0.00
(CNST)	
3 [C] 6 [ENTER↑] 4 [B] [f] [A] [f] [C]	30.00
[D]	0.00
[E] 9 [ENTER↑] 11 [E]	60.00

Best solution (uses least amount of *Follow Me* memory):

Keystrokes:	Outputs:
[A]	0.00
6 [ENTER↑] 4 [f] [A] 3 [C] [f] [C]	30.00
[D]	0.00
5 [ENTER↑] 8 [E]	39.00
9 [ENTER↑] 11 [E]	60.00

Example 2:

A company determines the retail price of its products by adding the fixed cost of assembly and distribution to a variable parts cost then multiplying by 2.7. The company sets the wholesale price at 50% of the retail price. Use *Follow Me* to determine the retail and wholesale prices for the parts cost list below.

PARTS COST LIST

PART #	PARTS COST
0001	$17.35
0002	$21.18
0003	$26.07
0004	$28.75
0005	$33.15

$$\text{Retail cost} = [\text{Parts} + \text{Fixed}] \times 2.7$$
$$\text{Wholesale cost} = 50\% \text{ of retail cost}$$
$$\text{Fixed cost} = \$25/\text{unit}$$

Keystrokes:	**Outputs:**	
Teach the sequence to the calculator and compute results for the first part #.		
[A] 17.35 [ENTER↑] 25 [C] [■] [A] 2.7 [C] [■]		
[C] [B] →	114.35	(Retail)
50 [C] [■] [E] →	57.17	(Wholesale)
[D] →	0.00	
Compute prices for other parts.		
21.18 [E] →	124.69	
[E] →	62.34	
26.07 [E] →	137.89	
[E] →	68.94	
28.75 [E] →	145.13	
[E] →	72.56	
33.15 [E] →	157.01	
[E] →	78.50	

Example 3:

Use *Follow Me* to help evaluate the following formula using the data below.

$$y = 0.75\ A\ e^{0.63t}$$

A	2.3	2.8	3.7	6.4
t	1.0	2.0	4.5	6.0

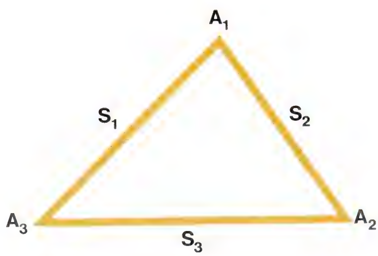
Keystrokes:	**Outputs:**
[A] 1 [ENTER↑] .63 [C] [■] [C] [B] [g] [e^x] 2.3	
[ENTER↑] .75 [C] [■] [C] [■] [C] →	3.24
[D] →	0.00
2.0 [E] [g] [e^x] 2.8 [E] →	7.40
4.5 [E] [g] [e^x] 3.7 [E] →	47.26
6.0 [E] [g] [e^x] 6.4 [E] →	210.32

Any keyboard function may be used during I/O halts.

NOTES

TRIANGLE SOLUTIONS

This program can be used to find the area, the dimensions of the sides (S_1, S_2, S_3) and the angles (A_1, A_2, A_3) of a triangle.



Simply key in three known values and press the corresponding user definable key. The calculator will successively display the values of the sides, the angles, and the area. The order of output is determined by the order of input. If input values are selected in a clockwise order around the triangle, the outputs will also follow a clockwise order around the triangle. The order is as follows:

First side input	(S_1)
Adjacent angle	(A_1)
Adjacent side	(S_2)
Adjacent angle	(A_2)
Adjacent side	(S_3)
Adjacent angle	(A_3)
Area	

After calculation has ended, the area will be in the display, S_1 in R_9, A_1 in R_A, S_2 in R_B, A_2 in R_C, S_3 in R_D, and A_3 in R_E.

Equations:

S_1, S_2, S_3 (all sides of triangle are known)

$$A_3 = 2 \cos^{-1} \sqrt{\frac{P(P - S_2)}{S_1 \, S_3}}$$

where $P = (S_1 + S_2 + S_3)/2$

$$A_2 = 2 \cos^{-1} \sqrt{\frac{P(P - S_1)}{S_2 S_3}}$$

$$A_1 = \cos^{-1} \left(-\cos (A_3 + A_2)\right)$$

A_3, S_1, A_1 (Two angles and the included side are known)

$$A_2 = \cos^{-1} \left(-\cos (A_3 + A_1) \right)$$

$$S_2 = S_1 \frac{\sin A_3}{\sin A_2}$$

$$S_3 = S_1 \cos A_3 + S_2 \cos A_2$$

S_1, A_1, A_2 (side and following two angles known)

$$A_3 = \cos^{-1} \left(-\cos (A_1 + A_2) \right)$$

Problem has been reduced to the A_3, S_1, A_1 configuration.

S_1, A_1, S_2 (Two sides and included angle are known)

$$S_3 = \sqrt{S_1^2 + S_2^2 - 2 S_1 S_2 \cos A_1}$$

The problem has been reduced to the S_1, S_2, S_3 configuration.

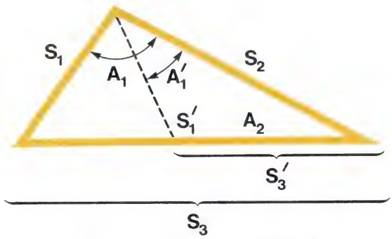
S_1, S_2, A_2 (Two sides and the adjacent angle known)

$$A_3 = \sin^{-1} \left[\frac{S_2}{S_1} \sin A_2 \right]^*$$

$$A_1 = \cos^{-1} \left[-\cos (A_2 + A_3) \right]$$

The problem has been reduced to the A_3, S_1, A_1 configuration.

*Note that two possible solutions exist if S_2 is greater than S_1 and A_3 does not equal 90°. Both possible answer sets are calculated.



Area = 1/2 S_1 S_3 sin A_3

Remarks:

Registers R_0 - R_6, R_{S0} - R_{S9} and I are available for user storage.

Angles must be in units corresponding to the angular mode of the machine. Degrees mode is set when the program is loaded.

Note that the triangle described by the program does not conform to standard triangle notation; i.e., A_1 is not opposite S_1.

Angles must be entered as decimals. The H.MS→ conversion can be used to convert degrees, minutes, and seconds to decimal degrees.

Accuracy of solution may degenerate for triangles containing extremely small angles.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	Load side 1 and side 2.			
2	Find applicable case in the list			
	below and input indicated			
	values:			
	All sides known	S_1	ENTER↑	S_1
		S_2	ENTER↑	S_2
		S_3	A	S_1, A_1, S_2...
	Two angles and included side			
	known	A_3	ENTER↑	A_3
		S_1	ENTER↑	S_1
		A_1	B	S_1, A_1, S_2...

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
	Two angles and adjacent side			
	known	S_1	ENTER↑	S_1
		A_1	ENTER↑	A_1
		A_2	C	S_1, A_1, S_2...
	Two sides and included angle			
	known	S_1	ENTER↑	S_1
		A_1	ENTER↑	A_1
		S_2	D	S_1, A_1, S_2...
	Two sides and adjacent angle			
	known	S_1	ENTER↑	S_1
		S_2	ENTER↑	S_2
		A_2	E	S_1, A_1, S_2...
3	After step 2, the values of the			
	sides and angles of the triangle			
	are successively displayed. The			
	first value output is the first			
	side input. The next five outputs			
	are the remaining angles and			
	sides. The last output is the			
	triangle's area. For the last case			
	(S_1, S_2, A_2), two possible			
	solutions may exist and both			
	will be output.			

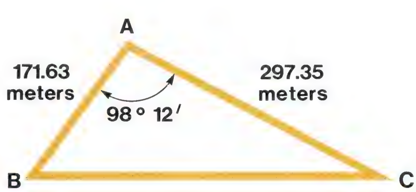
Example 1:

Find the angles and the area for the following triangle.

Keystrokes:	**Outputs:**	
2 **ENTER↑** 1 **ENTER↑** 2.75 **A** →	2.00 ***	
	129.84 ***	(A_1)
	1.00 ***	
	33.95 ***	(A_2)
	2.75 ***	
	16.21 ***	(A_3)
	0.77 ***	(Area)
RCL 9 →	2.00	
RCL **A** →	129.84	
RCL **B** →	1.00	
RCL **C** →	33.95	
RCL **D** →	2.75	
RCL **E** →	16.21	

Example 2:

A surveyor is to find the area and dimensions of a triangular land parcel. From point A, the distances to B and C are measured with an electronic distance meter. The angle between AB and AC is also measured. Find the area and other dimensions of the triangle.



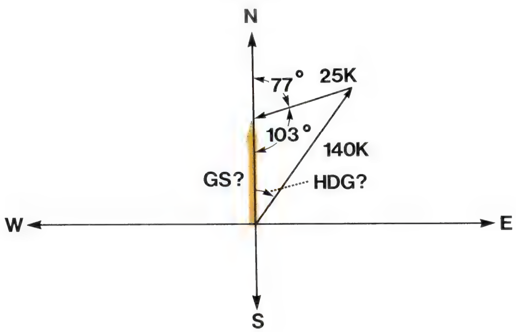
This is a side-angle-side problem where:

$$S_1 = 171.63,\ A_1 = 98°12' \text{ and } S_2 = 297.35.$$

Keystrokes:	Outputs:
171.63 ENTER↑ 98.12 f H.MS→	
297.35 D →	171.63 *** (AB)
	98.20 *** (∠A)
	297.35 *** (AC)
	27.83 *** (∠C)
	363.91 *** (CB)
	53.97 *** (∠B)
	25256.21 ***(Area)

Example 3:

A pilot wishes to fly due north. The wind is reported as 25 knots at 77°. Because winds are reported opposite to the direction they blow, this is interpreted as 77 + 180 or 257°. The true airspeed of the aircraft is 140 knots. What heading (HDG) should be flown? What is the ground speed (GS)?



By subtracting the wind direction from 180 (yielding an angle of 103°), the problem reduces to a S_1, S_2, A_2 triangle.

Keystrokes:	Outputs:
140 ENTER↑ 25 ENTER↑ 103 E →	140.00 *** (TAS)
	66.98 ***
	25.00 *** (Wind velocity)
	103.00 ***
	132.24 *** (GS)
	10.02 *** (HDG)
	1610.64 ***

Thus, the pilot should fly a heading 10.02° east of due north. His ground speed equals 132.24 knots.

VECTOR OPERATIONS

This program performs the basic vector operations of addition, cross product, and dot or scalar product. It also allows conversion between spherical and cartesian coordinates and can find the angle between two vectors.

Either two-dimensional or three-dimensional space may be selected using the f A keys. The machine is set in two-dimensional mode when the program is loaded. The first press of f A yields a display of 3.00 indicating three-dimensional space. Repeatedly pressing f A will yield alternate displays of 2.00 and 3.00 indicating the mode of the machine. Be sure the mode is correct before input of data.

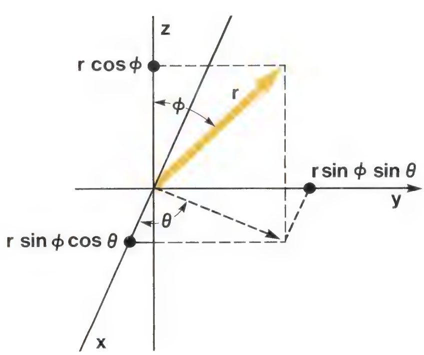
Another available option allows review of input values. Pressing f B causes a 1.00 to be displayed alternately indicating that the pause input mode is on or off. A print stack command is used to successively display the inputs in the following format:

Vector number (1.00 or 2.00)	T
ϕ (or $\pi \div 2$ for 2D vectors)	Z
θ	Y
r	X

Vector outputs are displayed in the following order:

POLAR FORM		RECTANGULAR FORM (S→C only)	
0.00	T	0.00	T
ϕ	Z	z	Z
θ	Y	y	Y
r	X	x	X

Equations:



Coordinate conversions:

$$x = r \sin \phi \cos \theta \qquad r = \sqrt{x^2 + y^2 + z^2}$$

$$y = r \sin \phi \sin \theta \qquad \theta = \tan^{-1} (y/x)$$

$$z = r \cos \phi \qquad \phi = \cos^{-1} \left(z/ \sqrt{x^2 + y^2 + z^2} \right)$$

Vector addition:

$$\vec{V}_1 + \vec{V}_2 = (x_1 + x_2) \vec{i} + (y_1 + y_2) \vec{j} + (z_1 + z_2) \vec{k}$$

Cross product:

$$\vec{V}_1 \times \vec{V}_2 = (y_1 z_2 - z_1 y_2) \vec{i} + (z_1 x_2 - x_1 z_2) \vec{j} + (x_1 y_2 - y_1 x_2) \vec{k}$$

Dot or scalar product:

$$\vec{V}_1 \cdot \vec{V}_2 = x_1 x_2 + y_1 y_2 + z_1 z_2$$

Angle between vectors:

$$\gamma = \cos^{-1} \frac{\vec{V}_1 \cdot \vec{V}_2}{|\vec{V}_1| \, |\vec{V}_2|}$$

Remarks:

Registers $R_0 - R_6$ and $R_{S0} - R_{S9}$ are available for user storage.

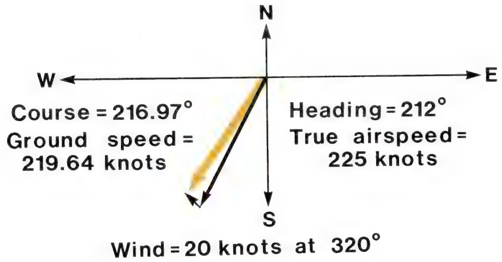
STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	Load side 1 and 2.			
2	Select mode for 2-dimensional			
	or 3-dimensional vectors.		f A	3.00/2.00
3	Optional: Select pause input			
	mode.		f B	1.00/0.00
4	If coordinate conversion			
	needed:			
	Spherical to Cartesian-go to			
	step 8.			
	Cartesian to spherical-go to			
	step 10.			

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
5	Input vectors one and two:			
	Co-latitude (skip for 2D)	(ϕ_1)	ENTER↑	(ϕ_1)
	Longitude	θ_1	ENTER↑	θ_1
	Magnitude	r_1	D	1.00
	Co-latitude (skip for 2D)	(ϕ_2)	ENTER↑	(ϕ_2)
	Longitude	θ_2	ENTER↑	θ_2
	Magnitude	r_2	E	2.00
6	Perform vector operation:			
	Add vectors		A	0, ϕ, θ, r
	Cross product		B	0, ϕ, θ, r
	Dot product		C	$\vec{V}_1 \cdot \vec{V}_2$, γ
7	For a new case go to steps 2, 3,			
	4 or 5.			
8	Input spherical coordinates:			
	(converts to Cartesian)			
	Co-latitude (skip for 2D)	(ϕ)	ENTER↑	(ϕ)
	Longitude	θ	ENTER↑	θ
	Magnitude	r	f D	x
9	For a new case go to steps 2, 3,			
	4 or 5.			
10	Input Cartesian coordinates			
	(converts to spherical)			
	z—distance (skip for 2D)	(z)	ENTER↑	(z)
	y—distance	y	ENTER↑	y
	x—distance	x	f E	r
11	For a new case go to steps 2, 3,			
	4 or 5.			

Example 1:

An aircraft flies a heading of 212 degrees at 225 knots. The wind is reported at 20 knots and 140 degrees (which translates to 20 knots and 320 degrees since

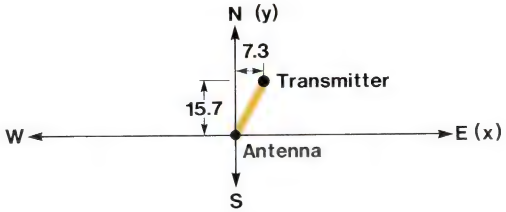
winds are reported opposite to the direction they blow). What is the course of the aircraft? What is the ground speed?



Keystrokes:	Outputs:	
f A f A	2.00	
212 ENTER↑ 225 D	1.00	
320 ENTER↑ 20 E	2.00	
A	0.00 ***	T
	90.00 ***	Z
	216.97 ***	Y (degrees)
	219.64 ***	X (knots)

Example 2:

A microwave antenna is to be pointed at a transmitter which is 15.7 kilometers north, 7.3 kilometers east and 0.76 kilometers below. Use the cartesian to spherical conversion to find the total distance and the direction to the transmitter.

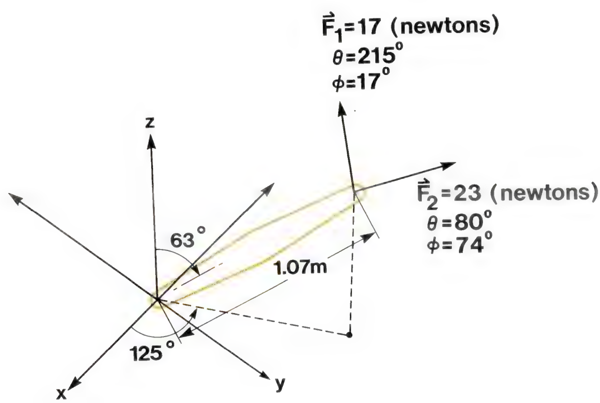


Keystrokes:	Outputs:
f A	3.00
.76 CHS ENTER↑ 15.7 ENTER↑	

7.3 f E	17.33	(distance)
h R↓	65.06	(θ from east)
h R↓	92.51	(ϕ from vertical)
h R↓	0.00	
h R↓	17.33	(back to distance)

Example 3:

What is the moment at the origin of the lever shown below? What is the component of force along the lever? What is the angle between the resultant of the force vectors and the lever?



Keystrokes: **Outputs:**

First, add $\vec{F}_1$ and $\vec{F}_2$

f A	3.00	(3D mode)
17 ENTER↑ 215 ENTER↑ 17 D	1.00	
74 ENTER↑ 80 ENTER↑ 23 E	2.00	
A	0.00 ***	T
	39.34 ***	Z
	90.70 ***	Y
	29.47 ***	X (newtons)

Keystrokes: **Outputs:**

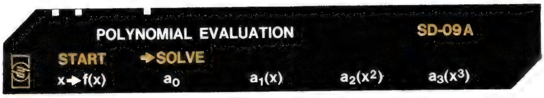
Take cross product for moment, $\vec{M} = \vec{r} \times \vec{F}$

E	2.00	
63 ENTER↑ 125 ENTER↑ 1.07 D	1.00	
B	0.00 ***	T
	124.34 ***	Z
	55.37 ***	Y
	18.02 ***	X

Take dot product to resolve force along the lever.

Keystrokes:	**Outputs:**	
63 ENTER↑ 125 ENTER↑ 1 D →	1.00	
C →	24.19 ***	(newtons)
	34.85 ***	(degrees)

POLYNOMIAL EVALUATION



This program may be used to find the roots of the following equations:

Cubic equation (3 roots)

$$f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 = 0$$

Quadratic equation (2 roots)

$$f(x) = a_0 + a_1x + a_2x^2 = 0$$

Linear equation (1 root)

$$f(x) = a_0 + a_1x = 0$$

where a_0, a_1, a_2 and a_3 are the polynomial coefficients input by the user. Both real and imaginary roots can be extracted. When imaginary roots are found, a –1. is displayed followed by imaginary and real parts. Real roots are displayed without the –1. indicator. Example 3 involves imaginary roots and should make this clear.

Polynomials may also be evaluated for arbitrary values of x. This is of aid in plotting polynomials and using data correlations based on polynomials. Example 2 demonstrates this type of use.

Equations:

Cubic Equation:

$$Q = \frac{3a_1 - a_2^2/a_3}{9a_3}$$

$$R = \frac{9a_2a_1/a_3 - 27a_0 - 2a_2^3/a_3^2}{54a_3}$$

$$S = \sqrt[3]{R + \sqrt{Q^3 + R^2}}$$

$$T = \sqrt[3]{R - \sqrt{Q^3 + R^2}}$$

If $$Q^3 + R^2 \geq 0,$$

then $$x_3 = S + T - \frac{a_2}{3a_3}$$

If $$Q^3 + R^2 < 0,$$

then $$x_3 = 2\sqrt{-Q}\cos\left[\frac{1}{3}\cos^{-1}(R/\sqrt{-Q^3})\right] - \frac{a_2}{3a_3}$$

After x_3 is found, synthetic division is performed to reduce the cubic equation to a quadratic equation.

$$a_2' = 1.00$$

$$a_1'/a_2' = x_3 + a_2/a_3$$

$$a_0'/a_2' = x_3(x_3 + a_2/a_3) + a_1/a_3$$

Quadratic equation:

$$x_1 = \begin{cases} -\dfrac{a_1}{2a_2} - \sqrt{(a_1/2a_2)^2 - (a_0/a_2)} & \text{If } -a_1/2a_2 < 0 \\ -\dfrac{a_1}{2a_2} + \sqrt{(a_1/2a_2)^2 - (a_0/a_2)} & \text{If } -a_1/2a_2 \geq 0 \end{cases}$$

$$x_2 = \frac{a_0}{a_2 x_1}$$

Linear equation:

$$x = -\frac{a_0}{a_1}$$

Remarks:

Registers R_0, $R_5 - R_9$, and $R_{S0} - R_{S9}$ are available for user storage.

Accuracy degenerates if the real root of the cubic equation is extremely small.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	Load side 1 and side 2.			
2	Initialize		f A	0.00
3	Input coefficients of Polynomial:			
	Constant	a_0	B	1.00
	x coefficient	a_1	C	2.00
	x^2 coefficient	a_2	D	3.00
	x^3 coefficient	a_3	E	4.00
4	To evaluate polynomial for			
	various values of x go to step 7.			
5	Find the roots of the polynomial.			
	(Imaginary roots will be output			
	in imaginary, real order preced-			
	ed by a negative one).		f B	roots
6	Go to step 8.			
7	Input x and see f(x)	x	A	f(x)
8	For a new case of same or high-			
	er degree, go to step 3 and			
	change appropriate coefficients.			
	For a lower degree go to step 2.			

Example 1:

A ball is thrown straight up at a velocity of 20 meters per second, from a height of 2 meters. At what time, neglecting air resistance, will it reach the ground? The acceleration of gravity is 9.81 meters per second. From physics:

$$f(t) = x = x_0 + v_0 t + \frac{1}{2} a t^2 = 0$$

$$= 2 + 20t + (-9.81/2)t^2 = 0$$

Keystrokes: **Outputs:**

f A ⟶ 0.00

2 B 20 C 9.81 ENTER↑		
2 ÷ CHS D f B →	4.18 ***	(seconds)
	–0.10 ***	(seconds)

The answer is 4.18 seconds. The second root of –0.10 is a legitimate root of the equation but is not relevant to this problem.

Example 2:

The standard heat of formation of ammonia (NH_3) is given as a function of Kelvin temperature by:

$$\Delta H_T^\circ = -9140 - 7.596\ T + 4.243 \times 10^{-3}\ T^2 - 0.742 \times 10^{-6}\ T^3\ \text{(cal)}$$

Determine the heat of formation for temperatures of 400 K, 600 K, and 800 K.

Keystrokes:	**Outputs:**	
f A →	0.00	
9140 CHS B 7.596 CHS C →	2.00	
4.243 EEX CHS 3 D .742		
CHS EEX CHS 6 E →	4.00	
400 A →	–11547.01	(cal)
600 A →	–12330.39	(cal)
800 A →	–12881.18	(cal)

Example 3:

Find the roots of the following equation.

$$x^3 - 4x^2 + 8x - 8 = 0$$

Keystrokes:	**Outputs:**	
f A 8 CHS B 8 C		
4 CHS D 1 E f B →	2.00 ***	(real root)
	–1.	(indicator)
	1.73 ***	(imaginary part)
	1.00 ***	(real part)

The real root is 2.00. The imaginary roots are 1.00 + 1.73i and 1.00 − 1.73i. The −1. (which is not followed by asterisks) indicates that the last two outputs will be imaginary and real parts rather than real roots.

3 × 3 MATRIX OPERATIONS

This program can be used to find the determinant or generate the inverse of a 3 × 3 matrix. It can also multiply a 3 × 3 matrix by a column matrix. By using the matrix inverse function in combination with the matrix multiply function, it is possible to solve three linear equations in three unknowns.

Equations:

$$\text{Matrix A} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix}$$

$$\text{Matrix D} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}$$

Determinant of matrix A

$$\begin{aligned} \text{Det} = {} & a_1 b_2 c_3 + b_1 c_2 a_3 + c_1 b_3 a_2 \\ & - c_1 b_2 a_3 - c_2 b_3 a_1 - c_3 a_2 b_1 \end{aligned}$$

Inverse of matrix A

$$A^{-1} = \begin{bmatrix} \alpha_1 & \beta_1 & \gamma_1 \\ \alpha_2 & \beta_2 & \gamma_2 \\ \alpha_3 & \beta_3 & \gamma_3 \end{bmatrix}$$

$$\begin{aligned} \alpha_1 &= (b_2\, c_3 - b_3\, c_2)/\text{Det} \\ \alpha_2 &= (a_3\, c_2 - a_2\, c_3)/\text{Det} \\ \alpha_3 &= (a_2\, b_3 - a_3\, b_2)/\text{Det} \\ \beta_1 &= (b_3\, c_1 - b_1\, c_3)/\text{Det} \\ \beta_2 &= (a_1\, c_3 - a_3\, c_1)/\text{Det} \\ \beta_3 &= (a_3\, b_1 - a_1\, b_3)/\text{Det} \end{aligned}$$

$$\gamma_1 = (b_1 c_2 - b_2 c_1)/\text{Det}$$
$$\gamma_2 = (a_2 c_1 - a_1 c_2)/\text{Det}$$
$$\gamma_3 = (a_1 b_2 - a_2 b_1)/\text{Det}$$

Matrix multiplication

$$A \cdot D = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}$$

$$= \begin{bmatrix} a_1d_1 + b_1d_2 + c_1d_3 \\ a_2d_1 + b_2d_2 + c_2d_3 \\ a_3d_1 + b_3d_2 + c_3d_3 \end{bmatrix}$$

Remarks:

During matrix inversion, A^{-1} replaces A in storage. If you wish to save matrix A, store it on a magnetic card before starting the inversion process.

Two by two matrix operations can be performed with this program (see example 2). A 2 × 2 matrix should be input in the following form:

$$A = \begin{bmatrix} a_1 & b_1 & 0 \\ a_2 & b_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The corresponding column vector is:

$$D = \begin{bmatrix} d_1 \\ d_2 \\ 0 \end{bmatrix}$$

If the determinant of a matrix is zero, the inverse cannot be found.

Registers R_{S0}–R_{S9} are available for user storage.

Matrices may be output at any time by pressing **E**. The order of output is $a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2, c_3, d_1, d_2, d_3$.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	Load side 1 and side 2.			
2	Input 3 × 3 matrix:			
	Column 1	a_1	ENTER↑	a_1
		a_2	ENTER↑	a_2
		a_3	A	a_3
	Column 2	b_1	ENTER↑	b_1
		b_2	ENTER↑	b_2
		b_3	B	b_3
	Column 3	c_1	ENTER↑	c_1
		c_2	ENTER↑	c_2
		c_3	C	c_3
3	For solution of simultaneous			
	equations or multiplication of			
	the 3 × 3 matrix by a column			
	matrix, input column matrix.	d_1	ENTER↑	d_1
		d_2	ENTER↑	d_2
		d_3	D	d_3
4	To find a determinant go to step			
	5. To find the inverse or solve a			
	3 × 3 system, go to step 8. To			
	perform multiplication, go to			
	step 10.			
5	Find the determinant of the			
	3 × 3 matrix.		f A	$\|A\|$
6	For a new case,go to step 2.			
	Change any or all of the columns			
	in step 3.			
7	If you wish to save the 3 × 3			
	matrix for future use, record it			
	on a magnetic card.			

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
8	Find the inverse.		f B	0.00
9	For a solution of a 3 × 3 system			
	go to step 10. For a new case go			
	to step 2. The original 3 × 3			
	matrix has been replaced in			
	storage by its 3 × 3 inverse.			
10	Multiply the 3 × 3 matrix by the			
	column matrix. (The resulting			
	column matrix is output in x, y, z			
	order).		f C	x, y, z
11	For multiplication by another			
	column matrix, perform step 3,			
	then press f C. For a new			
	case go to step 2.			

Example 1:

Find the determinant and inverse of the following matrix; then multiply by the column matrix.

$$\begin{bmatrix} 23 & 15 & 17 \\ 8 & 11 & -6 \\ 4 & 15 & 12 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Keystrokes:		**Outputs:**
23 ENTER↑ 8 ENTER↑ 4 A	→	4.00
15 ENTER↑ 11 ENTER↑ 15 B	→	15.00
17 ENTER↑ 6 CHS ENTER↑ 12 C	→	12.00
1 ENTER↑ 1 ENTER↑ 1 D	→	1.00

f A ⟶	4598.00	(determinant)
f B ⟶	0.00	(inverse found)
E ⟶	0.05 ***	(α_1)
	-0.03 ***	(α_2)
	0.02 ***	(α_3)
	0.02 ***	(β_1)
	0.05 ***	(β_2)
	-0.06 ***	(β_3)
	-0.06 ***	(γ_1)
	0.06 ***	(γ_2)
	0.03 ***	(γ_3)
	1.00 ***	(d_1)
	1.00 ***	(d_2)
	1.00 ***	(d_3)
	(results of multiplication)	
f C ⟶	4.349717270 -03 ***	
	0.08 ***	
	-0.02 ***	

Example 2:

Find the determinant and the inverse of the 2 × 2 matrix below. After the inverse has been found, multiply by the column matrix.

$$\begin{bmatrix} 14 & -8 \\ -8 & 12 \end{bmatrix} \quad \begin{bmatrix} 20 \\ 5 \end{bmatrix}$$

First transform the matrices to three dimensions as specified in the remarks section:

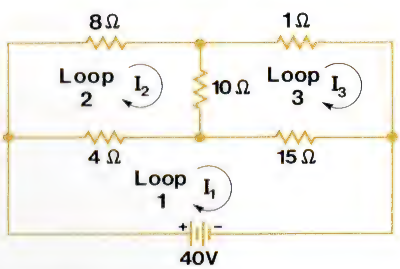
$$\begin{bmatrix} 14 & -8 & 0 \\ -8 & 12 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 20 \\ 5 \\ 0 \end{bmatrix}$$

Keystrokes:	**Outputs:**
14 ENTER↑ 8 CHS ENTER↑ 0 A ⟶	0.00
8 CHS ENTER↑ 12 ENTER↑ 0 B ⟶	0.00
0 ENTER↑ 0 ENTER↑ 1 C ⟶	1.00
20 ENTER↑ 5 ENTER↑ 0 D ⟶	0.00

f A	104.00	(determinant)
f B	0.00	(inverse has been found)
E	0.12 ***	(α_1)
	0.08 ***	(α_2)
	0.00 ***	(α_3)
	0.08 ***	(β_1)
	0.13 ***	(β_2)
	0.00 ***	(β_3)
	0.00 ***	(γ_1)
	0.00 ***	(γ_2)
	1.00 ***	(γ_3)
	20.00 ***	(d_1)
	5.00 ***	(d_2)
	0.00 ***	(d_3)
f C	2.69 ***	(results of multiplication)
	2.21 ***	
	0.00 ***	

Example 3:

Solve for the loop currents in the following circuit.



The three loop equations are:

Loop 1 $\quad 4I_1 - 4I_2 + 15\,I_1 - 15\,I_3 - 40 = 0$

Loop 2 $\quad 4\,I_2 - 4\,I_1 + 8\,I_2 + 10\,I_2 - 10\,I_3 = 0$

Loop 3 $\quad 10\,I_3 - 10\,I_2 + 1\,I_3 + 15\,I_3 - 15\,I_1 = 0$

or

$$19\,I_1 - 4\,I_2 - 15\,I_3 = 40$$
$$-4\,I_1 + 22\,I_2 - 10\,I_3 = 0$$
$$-15\,I_1 - 10\,I_2 + 26\,I_3 = 0$$

or in matrix form

$$\begin{bmatrix} 19 & -4 & -15 \\ -4 & 22 & -10 \\ -15 & -10 & 26 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} 40 \\ 0 \\ 0 \end{bmatrix}$$

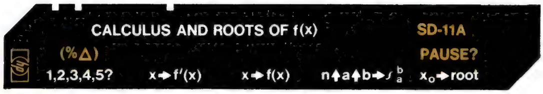
and

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} 19 & -4 & -15 \\ -4 & 22 & -10 \\ -15 & -10 & 26 \end{bmatrix}^{-1} \begin{bmatrix} 40 \\ 0 \\ 0 \end{bmatrix}$$

Keystrokes:	Outputs:	
19 ENTER↑ 4 CHS ENTER↑ 15 CHS A →	−15.00	
4 CHS ENTER↑ 22 ENTER↑ 10 CHS B →	−10.00	
15 CHS ENTER↑ 10 CHS ENTER↑ 26 C →	26.00	
40 ENTER↑ 0 ENTER↑ 0 D →	0.00	
f B →	0.00	(inverse has been found)
f C →	7.86 ***	(I_1)
	4.23 ***	(I_2)
	6.16 ***	(I_3)

NOTES

CALCULUS AND ROOTS OF f(X)



This program incorporates four routines for numerical analysis of user specified functions. Suppose figure 1 represents a known function of x called f(x).

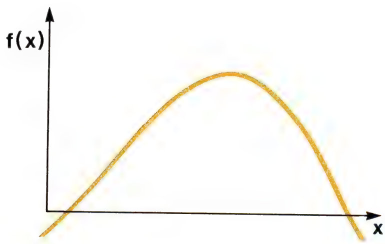


Figure 1

If the formula for f(x) can be keyed into program memory in less than 112 steps (including LBL and RTN), this program can be used to find the value of f(x) at any point x, the derivative of f(x) at any point x, the integral of f(x) over a specified interval and the real roots of f(x). There may be up to five different f(x) functions in program memory at one time. They must be labeled from 1 to 5. The function to be evaluated is selected by keying in 1, 2, 3, 4 or 5 and pressing **A**.

Only side 1 of *Calculus and Roots of f(x)* is used for the program. Side 2 of *Calculus and Roots of f(x)* has three functions recorded on it. These will be used in the example problems to show various applications of the program. You may wish to record functions you use frequently on blank magnetic cards. Once recorded, the functions can be linked to *Calculus and Roots of f(x)* by the following sequence of operations:

1. Load side 1 of *Calculus and Roots of f(x)*.
2. Press **GTO** **.** **1** **1** **2**.
3. Press **g** **MERGE**.
4. Load your magnetic card.

Once a function is defined and selected, keying in a value of x and pressing the **C** key will result in the evaluation of f(x) (see figure 2).

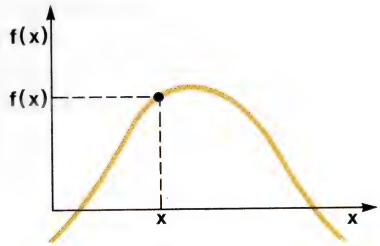


Figure 2

Similarly, the value of the slope of f(x) at a particular point x can be calculated by keying in x and pressing the **B** key (see figure 3). The slope of f(x) is determined using an approximation to the differential:

$$f'(x) = \frac{f(x + \Delta x/2) - f(x - \Delta x/2)}{\Delta x}$$

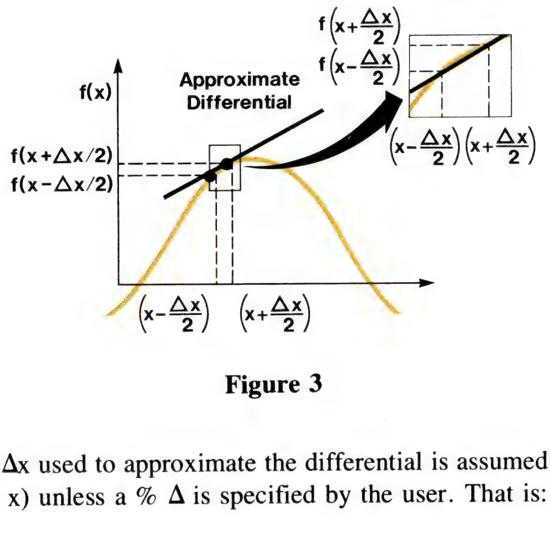


Figure 3

The value of Δx used to approximate the differential is assumed to be 0.01% of x ($10^{-4} \times x$) unless a % Δ is specified by the user. That is:

$$\Delta x = \frac{\%\Delta}{100} \cdot x$$

In the special case where $x = 0$, Δx is set equal to $\%\Delta$.

For most applications, the assumed value of 0.01% should be adequate. In some cases more accurate results can be obtained using a smaller value of

%Δ. However, care must be taken to assure that the calculator can accurately resolve the difference between $f(x - \Delta x/2)$ and $f(x + \Delta x/2)$.

The **D** key may be used to approximate the integral or area under a curve.

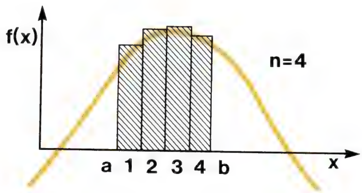


Figure 4

You specify the end points of the interval (a and b) and the number of rectangles (n) the interval should be broken into (see figure 4). The calculator computes the sum of the areas of the rectangles. The more rectangles used the closer this value is to the actual area under the curve. However, more rectangles mean more computation time. Experience with a particular function should lead to a balance between accuracy and execution time.

Root finders are used to solve equations which are difficult or impossible to solve explicitly. An example of such an equation is

$$f(x) = \ln x + 3x - 10.8074 = 0$$

which is solved in example 4.

The root finder incorporated in this program uses a secant method of approximation. You must supply the routine with an initial guess of the root. Based on this guess, it will attempt to make better and better approximations of the root by the following formula:

$$x_{i+1} = x_i - f(x_i)\left[\frac{(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}\right]$$

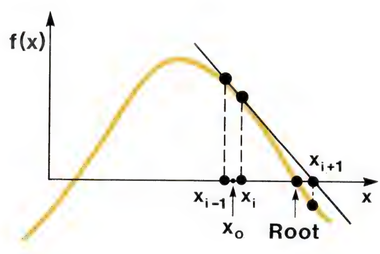


Figure 5

The display is automatically set to fix mode during the root finder portion of the program. When the last approximation is accurate to the number of places specified by the display setting of the calculator, the routine halts and displays the root.

Since the root finder starts its search based on your guess, care should be exercised in guess selection. A bad guess will cause long execution times and could result in a machine status error halt (overflow, division by zero, log of a negative number, etc.). If this happens, simply try another guess. Practice will make the pitfalls more obvious and easier to avoid.

A special feature of the iterative routine is the pause function. This feature allows the program to pause at one point in each iteration to display the current approximation of the root. The pause option may be turned off and on by pressing f E. The pause allows you to watch the routine converge (or diverge) without interrupting the program. This can be a helpful tool when the iterative routine fails to converge. By watching each successive approximation of the root, the reasons for failure of convergence can usually be determined.

Remarks:

The value of x is stored in R_0 by the program. It is also in the X register when control transfers to the function subroutine.

Registers R_1-R_8, and R_{S0}-R_{S9} are available for use in f(x) or for other user storage.

User-specified functions may use one level of subroutine nesting.

The secant method does not guarantee convergence to a root.

Given one guess, the root finder will find, at most, one root of an equation. Other real roots, if they exist, may be found by modifying the initial guess.

In order to compute $f'(x)$, the function $f(x)$ must be continuous on the interval $(x + \Delta x/2,\ x - \Delta x/2)$.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	Load side 1.			
2	Load subroutine(s) (either key			
	them in or link from program			
	step 112).			
3	Select function label number.	i(1-5)	A	i
4	Store any constants necessary			
	to subroutine(s) loaded in			
	step 2.			

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
5	For differentiation, go to step 6.			
	For evaluation of a function, go			
	to step 9. For integration of a			
	function, go to step 11. To find a			
	root, go to step 15.			
6	Optional: Key in percent delta.	%Δ	f A	%Δ
7	Key in x and calculate derivative			
	at x.	x	B	$f_i'(x)$
8	For new x, go to step 7. For a			
	new case, go to step 2, 3, 4, 5			
	or 6.			
9	Key in x and evaluate function.	x	C	$f_i(x)$
10	For new x, go to step 9. For a			
	new case, go to step 2, 3, 4, or 5.			
11	Input the number of intervals.	n	ENTER↑	n
12	Input the lower limit.	a	ENTER↑	a
13	Input the upper limit and			
	calculate the integral.	b	D	$\int f_i(x)\,dx$
14	For new limits or interval, go to			
	step 11. For a new case, go to			
	step 2, 3, 4 or 5.			
15	Optional: Key in percent delta.	%Δ	f A	%Δ
16	Optional: Toggle pause			
	mode.		f E	1.00/0.00
17	Key in guess and calculate root.	GUESS	E	x
18	For a new guess go to step 17.			
	For a new case go to step 2, 3,			
	4 or 5.			

Example 1:

Numerical integration provides the only solution to the complete elliptic integral of the first kind:

$$u = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - K^2 \sin\theta^2}}$$

Find the value of u for limits of integration of 0.0 to $\pi/2$. Let K be 0.5 and store it in register 1 for access by the program. Use 3 and then 10 for the number of intervals. The formula for the integral is recorded under label three on side two of the magnetic card. If either example 2 or example 3 has just been run, skip the first three lines under keystrokes.

Keystrokes:	Outputs:
Load side 1 only	
GTO [.] 112 g [MERGE]	
Load side 2	
Select label 3	
3 A ——→	3.00
0.50 STO [1] ——→	0.50
Integrate using 3 intervals	
[DSP] [9] 3 ENTER↑ 0 ENTER↑	
h [π] 2 ÷ D ——→	1.685750251
Integrate using 10 intervals	
10 ENTER↑ 0 ENTER↑ h [π] 2 ÷ D ——→	1.685750355

Example 2:

In the design of gear teeth, it is frequently necessary to calculate x for a given value of the involute:

$$INV(x) = \tan x - x$$

or restated

$$f(x) = \tan x - x - INV(x) = 0$$

If the involute of x is 0.0049819, what is x?

This problem requires an iterative solution since the equation cannot be explicitly solved for x. Use 0.21 radians as your initial guess. The equation for f(x) is recorded under label 2 on side 2 of the magnetic card. Use the pause

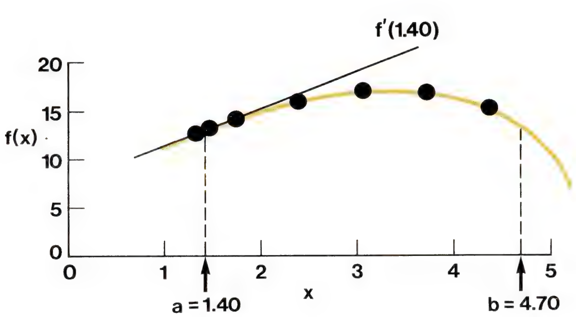
feature to watch the routine converge. Skip the first three lines under keystrokes if Example 1 or 3 has been run. Store the involute (.0049819) in R_2 for access by the function.

Keystrokes:	**Outputs:**
Load side 1 only	
GTO · 112 g MERGE	
Load side 2	
Select label 2	
2 A	2.00
Set pause	
DSP 2 f E	1.00
.0049819 STO 2 .21 E	"0.25"
	"0.24"
	"0.24"
	0.24 (rad)

Example 3:

In many instances, a function is represented graphically. This program can be of use in integration and, in some cases, differentiation of such graphs. Label 1 of side 2 of the prerecorded magnetic card is designed for this purpose. It returns x values to the display. You must find f(x) from the graph, key it in and press R/S.

For the function below find the integral from a to b using 5 intervals. Then find the derivative at a, using 10% for %Δ. After the problem is complete, return %Δ to 0.01%.



If either Example 1 or Example 2 was run previously, skip the first three lines under keystrokes.

Keystrokes:	**Outputs:**	
Load side 1 only		
GTO . 112 g MERGE		
Load side 2		
Select Label 1		
1 A →	1.00	
Key in integration limits and return first x value		
5 ENTER↑ 1.40 ENTER↑ 4.70 D →	1.73	(x)
From the graph, f(x) at x = 1.73 equals 14.2. Key 14.2 in and press R/S. The next value of x will be displayed.		
14.2 R/S →	2.39	
f (2.39) = 16		
16 R/S →	3.05	
f (3.05) = 17		
17 R/S →	3.71	
f (3.71) = 16.9		
16.9 R/S →	4.37	
f (4.37) = 15.3		
15.3 R/S →	52.40	(Answer)
To find the derivative at point a		
10 f A 1.40 B →	1.33	$\left(x - \frac{\Delta x}{2}\right)$
f (1.33) = 12.7		
12.7 R/S →	1.47	$\left(x + \frac{\Delta x}{2}\right)$
f (1.47) = 13.3		
13.3 R/S →	4.29	(Slope)
Return %Δ to 0.01%		
.01 f A →	0.01	

Example 4:

Find the root of $\ln x + 3x - 10.8074 = 0$. Determine the slope at the root.

This equation is not recorded on the magnetic card. It must be manually keyed into program memory starting at step 112. Use R_1 to store the 3 and R_2 to store 10.8074.

Keystrokes:	**Outputs:**
Load side 1 only	
GTO . 112	
Switch to W/PRGM →	112 35 22
f LBL 1 →	31 25 01

f LN →	114 31 52	(lnx)
RCL 1 →	115 34 01	
RCL 0 →	116 34 00	
× →	117 71	
+ →	118 61	(lnx + 3x)
RCL 2 →	119 34 02	
− →	120 51	(lnx + 3x − 10.8074)
h RTN →	121 35 22	
Switch to Run		
Select LBL 1		
1 A →	1.00	
3 STO 1 →	3.00	
10.8074 STO 2 →	10.81	
Make a guess of 5.0		
5 E →	3.21	(ROOT)
Find the derivative		
B →	3.31	f′ (3.21)

NOTES

ENGLISH-SI CONVERSIONS

This card provides the more common conversions between English and SI (metric) units. Side one of the card provides length, volume, force and mass conversions. Side two provides temperature, energy, pressure, density and power conversions. Only one side of the card may be loaded into program memory at any time.

Conversion Factors:

Side 1 of magnetic card

1 inch (in) = 25.4* millimeters (mm)
1 foot (ft) = 0.3048* meters (m)
1 U.S. liquid gallon (gal) = 3.785411784* liters (ℓ)
1 pound force avoirdupois (lbf) = 4.448221615 newtons (N)
1 pound mass avoirdupois (lbm) = 0.45359237* kilograms (kg)

Side 2 of magnetic card

Degrees Fahrenheit (°F) are related to degrees Celsius (°C) by the following formula:

$$°C = (°F - 32)/1.8$$

1 International Steam Table British thermal unit (Btu) = 1055.055853 joules (J)
1 pound per square inch (psi) = 6894.7572 newtons/square meters (N/m^2)
1 pound per cubic foot (lb/ft^3) = 16.018463 kilograms per cubic meter (kg/m^3)
1 horsepower (550 ft-lbf/sec) = 745.69987 watts (W)

Remarks:

Only one side of the card may be in program memory at a time.

All data registers (R_0 – I) are available for user storage. The T register of the operational stack is lost during conversions. The LAST X register contains the input value for all conversions except temperature conversions.

*By definition.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	For length, volume, force or			
	mass conversion, load side 1.			
	For temperature, energy, pres-			
	sure, density, or power conver-			
	sion, go to step 4.			
2	To convert inches to millimeters	in	A	mm
	or millimeters to inches	mm	■ A	in
	or feet to meters	ft	B	m
	or meters to feet	m	■ B	ft
	or gallons to liters	gal	C	ℓ
	or liters to gallons	ℓ	■ C	gal
	or pounds to newtons	lbf	D	N
	or newtons to pounds	N	■ D	lbf
	or pounds to kilograms	lbm	E	kg
	or kilograms to pounds	kg	■ E	lbm
3	For a new case, go to step 2.			
4	Load side 2.			
5	To convert Fahrenheit to Celsius	°F	A	°C
	or Celsius to Fahrenheit	°C	■ A	°F
	or Btu to joules	Btu	B	J
	or joules to Btu	J	■ B	Btu
	or psi to N/m^2	psi	C	N/m^2
	or N/m^2 to psi	N/m^2	■ C	psi
	or lb/ft^3 to kg/m^3	lb/ft^3	D	kg/m^3
	or kg/m^3 to lb/ft^3	kg/m^3	■ D	lb/ft^3
	or horsepower to watts	hp	E	W
	or watts to horsepower	W	■ E	hp
6	For a new case, go to step 5.			

Example 1:

Convert ⅜ of an inch to millimeters and round to an integer value.

Keystrokes:	**Output:**	
Load side one		
3 ENTER↑ 8 ÷ A →	9.53	(mm)
DSP 0 f RND →	10.	(mm)
DSP 2 →	10.00	(mm)

Example 2:

Convert 212°F to °C. Convert 0°C to °F.

Keystrokes:	**Outputs:**
Load side two	
212 A →	100.00
0 f A →	32.00

Example 3:

Convert 75 Btu/hr-ft^2 to joules/hr-m^2. (Since ft^2 is in the denominator, the sense of the conversion is reversed.)

Keystrokes:	**Output:**	
Side 1		
75 f B f B →	807.29	(Btu/hr-m^2)
Side 2		
B →	851739.50	(J/hr-m^2)

Example 4:

Convert six pounds per gallon to kilograms per liter.

Keystrokes:	**Outputs:**	
Side 1		
6 E f C →	0.72	(kg/ℓ)

NOTES

ARITHMETIC TEACHER

Preschool and elementary school students may use this program to help them learn addition, subtraction, multiplication, and division. The program generates and displays problems in the following form:

x .y

Where x is one variable and y is the other variable. The child mentally computes the answer (x + y, x − y, x × y, or x ÷ y depending on the lesson), keys it in, and presses the answer key E . If the answer is correct, the calculator poses a new problem. If the answer is incorrect, the calculator returns the problem until a correct response is given.

One lesson consists of 20 problems. After problem 20, the calculator outputs number correct, number tried, and percent correct.

As the child progresses, the maximum size of the numbers, n_{max}, may be modified. For example, keying in 3 and pressing f B would set the maximum number size to 3 for addition and multiplication, 3 + 3 for subtraction, and 3^2 for division. For more advanced students, n_{max} might be set to 15. If the value is not specified by the user, the program assumes a value of 9.

Remarks:

The type of problem to be solved (+, −, ×, ÷) can be changed at any time during the lesson. When the problem type is selected, a code number is displayed for a moment before a new problem is posed. The digit 1 indicates addition, 2 indicates subtraction, 3 indicates multiplication, and 4 indicates division.

If the student realizes that a wrong answer has been keyed in before the E key is pressed, the h R↓ keys can be used to eliminate the error and return the problem to the display.

Any attempt to use the calculator to solve the problem will result in an error necessitating a restart of the program.

The number generator incorporated in this program will always give the same sequence of numbers unless n_{max} is changed or a "seed" is input. The seed can be any number between 0 and 1. To input a seed, simply key it in and press f E.

Registers R_0 - R_6 and R_{S0} - R_{S9} are available for user storage.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	Load side 1 and side 2.			
2	Start program.		f A	0.00
3	Optional: Input a seed (any			
	number between 0 and 1).*	SEED	f E	0.00
4	Optional: Select maximum			
	number size (default is 9).	^{n}max	f B	0.00
5	Optional: Select print lesson			
	mode.		f C	1.00/0.00
6	Select arithmetic mode:* *			
	Addition		A	problem
	Subtraction		B	problem
	Multiplication		C	problem
	Division		D	problem
7	Let student key in answer and			
	press E.	answer	E	problem
8	Repeat step 7 for 20 problems.			
	After problem 20 the calculator			
	will output number correct,			
	number attempted and %			
	correct.			
9	For another session go to step 7.			
	To change arithmetic mode go			
	to step 6. To select print lesson			
	mode go to step 5. To sleect a			
	new maximum number size go			
	to step 4.			

* See page L13-01 for description of algorithm and comments on optional seed selection.

* After an arithmetic mode is selected a code is output to indicate which mode was set: 1 addition, 2 subtraction, 3 multiplication and 4 division.

Example 1:

A child is to practice multiplication of the numbers one through eight.

Keystrokes:		Outputs:
f A	→	0.00
Select maximum number size of 8.		
8 f B	→	8.0 ***
Select lesson type		
C	→	3.0 ***
		6.8
48 E	→	1.4
4 E	→	7.3
21 E	→	8.8
64 E	→	7.7
49 E	→	7.4
28 E	→	7.6
40 E	→	
45 E	→	
42 E	→	4.2
8 E	→	8.6
48 E	→	8.8
64 E	→	8.7
56 E	→	8.6
48 E	→	5.8
40 E	→	6.7
40 E	→	
42 E	→	5.8
40 E	→	8.4
32 E	→	4.6
24 E	→	7.4
28 E	→	4.4
16 E	→	4.7
28 E	→	18.0 ***
		20.
		90.0 ***

The calculator displays the first problem of the next set.

Example 2:

The child of example 1 now wishes to practice division for numbers 1 through 10.

Keystrokes:	Outputs:
10 f B	10.0 ***
D	4.0 ***
	30.06
5 E	70.07
10 E	30.06
5 E	28.04
7 E	32.08
4 E	6.06
1 E	80.10
8 E	40.04
10 E	16.04
4 E	80.08
10 E	70.10
7 E	80.08
10 E	42.07
6 E	81.09
9 E	7.07
1 E	10.05
2 E	60.06
6 E	
10 E	56.08
7 E	56.07
8 E	70.10
7 E	19.00 ***
	20.
	95.00 ***

MOON ROCKET LANDER

Imagine for a moment the difficulties involved in landing a rocket on the moon with a strictly limited fuel supply. You're coming down tail-first, freefalling toward a hard rock surface. You'll have to ignite your rockets to slow your descent; but if you burn too much too soon, you'll run out of fuel 100 feet up, and then you'll have nothing to look forward to but cold eternal moon dust coming faster every second. The object, clearly, is to space your burns just right so that you will alight on the moon's surface with no downward velocity.

The game starts off with the rocket descending at a velocity of 50 feet/second from a height of 500 feet. The velocity and altitude are shown in a combined display as –50.0500, the altitude appearing to the right of the decimal point and the velocity to the left, with a negative sign on the velocity to indicate downward motion. Then the remaining fuel is displayed and a rocket fire count down begins "3", "2", "1", "0",. Exactly at zero you may key in a fuel burn. You only have one second, so be ready. A zero burn, which is very common, is accomplished by doing nothing. However, if you miss the one second "fire window" and then try to key in a burn, your engine will die and you will have to restart by pressing **B**. This automatically uses 5 fuel units and gives no thrust. After a burn the sequence is repeated unless:

1. You have successfully landed—flashing zeros.
2. You have smashed into the lunar surface—flashing crash velocity.

You must take care, however, not to burn more fuel than you have; for if you do you will free-fall to your doom! The final velocity shown will be your inpact velocity (generally rather high). You have 60 units of fuel initially.

Equations:

We don't want to get too specific, because that would spoil the fun of the game; but rest assured that the program is solidly based on some old friends from Newtonian physics:

$$x = x_0 + v_0 t + \frac{1}{2} a t^2 \qquad v = v_0 + at \qquad v^2 = v_0^2 + 2ax$$

where x, v, a, and t are distance, velocity, acceleration, and time.

Remarks:

Only integer values for fuel burn are allowed.

R/S can be used to stop *Moon Rocket Lander* at any time.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	Load side 1.			
2	Assume manual control.		**A**	"V.ALT"
				"FUEL"
				"3"
				"2"
				"1"
3	Key in burn*.	BURN		"V.ALT"
				"FUEL"
				"3"
				"2"
				"1"
4	Go to step 3 until you land			
	(flashing zeros) or crash (flash-			
	ing impact velocity).			
5	If you survived last landing			
	attempt, go to step 2 for			
	another try.			
	*If you miss the burn window and			
	flameout, press **B** for a new			
	engine start.		**B**	

DIAGNOSTIC PROGRAM

This program can be used to test the calculator and diagnose calculator malfunctions. Simply insert the card and press **A**. After approximately two seconds, the calculator should pause displaying:

57.0

If the calculator does not pause with this number, there is a malfunction in executing and returning from a subroutine, finding Label 0, program storage, the display, the magnetic card, the PAUSE command or the card reader. After the pause, the calculator should continue to run about one-and-one-half minutes more and then print the three lines shown:

−888.9−90
−8.889−88
−8.888888888−88

This output indicates that printing and display formatting are working correctly. If the calculator stops before displaying ***−8.888888888−88***, a code number corresponding to a function or operation malfunction will be displayed. For instance, if the calculator stopped with ***36.0*** in the display, an error in tangent or arctangent would be indicated. The sole exception is a failure in primary register 0. The calculator will stop execution of the program with the erroneous contents of R_0 displayed.

DIAGNOSTIC CODES

Function or Operation or Register Indicated	Code
STO i, RCL i, R_0, GTO 0, LBL 0, $x=y$, $x \neq y$	0
ISZ I, R_1	1
R_2	2
R_3	3
R_4	4
R_5	5
R_6	6
R_7	7
R_8	8
R_9	9
R_{S0}	10
R_{S1}	11
R_{S2}	12

Remarks:

If this program runs correctly, it strongly suggests that the calculator is operating correctly. However, the diagnosis is by no means complete or exhaustive. The diagnostic can be made to repetitively loop by changing step 224 from "R/S" to "GTO A". This may aid in detection of intermittent failures. The program relies on the status of the flags to be correctly set by the card. If a flag error occurs, re-insert the diagnostic card and verify repeatability of failure.

ERROR CODES

Malfunction	Code	Malfunction	Code
R_1	1	y^x, LAST x, 1/x	30
R_2	2	$\sqrt{x}$, x^2	31
R_3	3	LN, e^x	32
R_4	4	LOG, 10^x	33
R_5	5	→H.MS, H.MS→, RND	34
R_6	6	→P, →R	35
R_7	7	TAN, TAN^{-1}	36
R_8	8	COS, COS^{-1}	37
R_9	9	DEG, SIN, SIN^{-1}	38
R_{S0}	10	FLAG 2, test cleared	39
R_{S1}	11	FLAG 1, set; LBL9	40
R_{S2}	12	FLAG 2, set; LBL8	41
R_{S3}	13	FLAG 0, clear	42
R_{S4}	14	FLAG 3, test cleared	43
R_{S5}	15	FLAG 0, set by card; LBL7	44
R_{S6}	16	FLAG 3, set by card; LBL6	45
R_{S7}	17	FLAG 1, cleared by card	46
R_{S8}	18	FLAG 2, cleared by card	47
R_{S9}	19	$x>0$, true; LBL4	48
R_A	20	$x<0$, false	49
R_B	21	$x=0$, false	50
R_C	22	$x\neq0$, true; LBL3	51
R_D	23	I-REGISTER	52
R_E	24	$x\leq y$, true; LBL1	53
EEX, %	25	$x=y$, false	54
D→R, R→D	26	$x>y$, false	55
FRC, INT	27	ENTER↑, R↓, R↑, x⇄y, STACK (X, Y, Z, T)	56
×, ÷	28	Subroutine execution and return, CLREG,	see text
+, −	29	P⇄S; LBL0	

Function or Operation or Register Indicated	Code
Flag 3, off	48
Flag 0, on	49
Flag 1, on	50
Flag 2, on	51
Flag 3, on	52

Remarks:

If this program runs correctly, it strongly suggests that the calculator is operating correctly. However, the diagnostic is by no means complete or exhaustive.

All data storage registers are used.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	Enter program			
2	Start diagnostic		A	-7.777777770-77
3	See documentation for descrip-			
	tion of outputs.			

PROGRAM LISTINGS* AND PROGRAMMING TECHNIQUES

Program	Page
1. Moving Average	L01-01
Comparisons	
2. Tabulator	L02-01
Decrement and Skip on Zero (DSZI)	
Loop in Combination with Indirect Recall (RCLi)	
3. Curve Fitting	L03-01
Primary Exchange Secondary Registers	
4. Calendar Functions	L04-01
Multiple Storage In Registers	
5. Annuities and Compound Amounts	L05-01
Interchangeable Solutions	
6. Follow Me	L06-01
Indirect GTO	
7. Triangle Solutions	L07-01
Variable Input	
8. Vector Operations	L08-01
Flag Set, Clear and Test—Command	
Clearing Flags	
9. Polynomial Evaluation	L09-01
Flag Set, Clear and Test—Test	
Clearing Flags	
10. Matrix Operations	L10-01
Subroutines and Indirect Recalls	
11. Calculus and Roots of f(x)	L11-01
Iterative Test and Loop	
12. Unit Conversions	L12-01
13. Arithmetic Teacher	L13-01
Pseudorandom Number Generator	
14. Moon Rocket Lander	L14-01
15. Diagnostic	L15-01

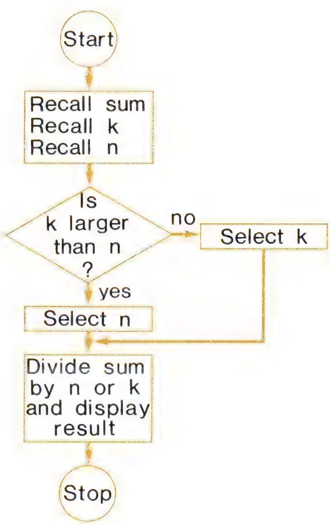
*Keycodes for program steps may be found in Appendix E of your Owner's Handbook.

COMPARISON

Subroutine D of *Moving Average* computes the moving average when the **D** key is pressed from the keyboard.

LBL D
RCL 0
RCL E
RCL D
x≤y?
x⇄y
R↓
÷
RTN

Generally, the average is calculated based on the summation of input values, Σ (stored in R_0) and the requested number of units, n (stored in R_D) in the moving average. However, if less than n values have been input, the average must be calculated based on the current number of inputs (k). The value of k is stored in R_E. The flowchart for this calculation might look like this:



Subroutine D begins by recalling the sum from R_0, k from R_E and n from R_D. After these recalls the operational stack is as follows:

Unknown value	T
Sum	Z
k	Y
n	X

The comparison step $x \leq y$ (if x is less than or equal to y) causes program execution to *skip* the next step when the conditions of the comparison are *not met*. If the conditions of the comparison are met, the *following step is executed*. This is the "*DO* if *TRUE*" rule. For instance, if $k = y = 15$ and $n = x = 6$ the comparison would be true or satisified (since x is less than y) and the next step, **x⇄y** (x exchange y), would be executed. If k were less than 6, say 4, the **x⇄y** command would be skipped. The stack contents for both cases are shown below:

BEFORE COMPARISON

Unknown value	T	Unknown value	T
Sum	Z	Sum	Z
15	Y	4	Y
6	X	6	X

AFTER COMPARISON AND NEXT STEP

Unknown value		T	Unknown value		T
Sum		Z	Sum		Z
6	switched	Y	4	not switched	Y
15		X	6		X

The next step rolls the stack down removing the unwanted value from the X-register.

15 (Unwanted value)	T	6 (Unwanted value)	T
Unknown value	Z	Unknown value	Z
Sum	Y	Sum	Y
6	X	4	X

The last step divides the sum by the value in the X-register to complete the calculation.

Moving Average

Step	Key	Comments
001	*LBLa	Clear registers.
002	CLRG	
003	P⇄S	
004	CLRG	
005	1	If 1 ≤ n ≤ 22 continue, otherwise go to label 1.
006	X>Y?	
007	GTO1	
008	CLX	
009	2	
010	2	
011	X⇄Y	
012	X>Y?	
013	GTO1	
014	STOD	Store n in R_D and (n + n/100) in R_I.
015	1	
016	%	
017	+	
018	STOI	
019	INT	
020	RTN	
021	*LBL1	Flash input error.
022	R↓	
023	*LBL4	
024	PSE	
025	GTO4	
026	*LBLA	Increment k by one. Print space, k, and input if flag 0 is set.
027	F0?	
028	SPC	
029	RCLE	
030	1	
031	+	
032	F0?	
033	PRTX	
034	X⇄Y	
035	F0?	
036	PRTX	
037	RCLi	Remove oldest value from sum and add input.
038	ST-0	
039	X⇄Y	
040	STOi	
041	ST+0	
042	R↓	Store k.
043	X⇄Y	
044	STOE	
045	RCLD	If n ≤ k, GTO 0 and calculate average.
046	X≤Y?	
047	GSB0	
048	DSZI	If I is not zero, GTO 5 for display
049	GTO5	
050	RCLI	Reset index for another loop.
051	1	
052	0	
053	1	
054	×	
055	STOI	
056	*LBL5	Display average or n.
057	R↓	If print mode is off pause for display of n.
058	RTN	
059	*LBL6	
060	X⇄Y	
061	F0?	
062	GTO0	
063	PSE	
064	*LBL0	Compute average.
065	RCL0	
066	RCLD	
067	÷	
068	ENT↑	Output and set for display.
069	F0?	
070	PRTX	
071	RTN	
072	*LBLB	Write data.
073	WDTA	
074	RTN	
075	*LBLb	Print/pause mode toggle.
076	F0?	
077	GTO6	
078	1	
079	SF0	
080	RTN	
081	*LBL6	
082	0	
083	CF0	
084	RTN	
085	*LBLC	Output values in newest to oldest order.
086	SPC	
087	0	
088	*LBL3	
089	RCLD	
090	X=Y?	
091	RTN	
092	1	
093	%	
094	+	
095	RCLI	
096	X=Y?	
097	FRC	
098	STOI	
099	ISZI	
100	RCLi	
101	PRTX	
102	R↑	
103	1	
104	+	
105	GTO3	
106	*LBLD	Compute average at any time.
107	RCL0	
108	RCLE	
109	RCLD	
110	X≤Y?	
111	X⇄Y	
112	R↓	

REGISTERS

0	1	2	3	4	5	6	7	8	9
Σ	used	used	used	used	used	used	used	used	used
S0	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**
used	used	used	used	used	used	used	used	used	used

A	B	C	D	E	I
used	used	used	n	k	control

```
113    ÷      -24
114    RTN     24
115    R/S     51
```

LABELS					FLAGS
A x→"k," Avg	B W DATA	C →VAL	D →AVG	E	0 print
a n	b P?	c	d	e	1
0 used	1 error	2	3 print	4 error	2
5 display	6	7	8	9	3

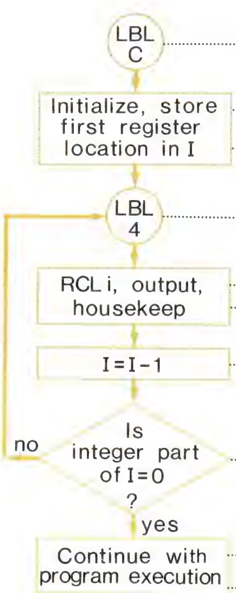
SET STATUS							
FLAGS	ON	OFF	TRIG		DISP	
0	☐	☒	DEG	☒	FIX	☒
1	☐	☒	GRAD	☐	SCI	☐
2	☐	☒	RAD	☐	ENG	☐
3	☐	☒			n	2

DECREMENT AND SKIP ON ZERO (DSZI) LOOP IN COMBINATION WITH INDIRECT RECALL (RCLi)

One of the most powerful features of your calculator is its ability to do indirect recalls. That is, recall a register which is specified by a value stored in the I register. For instance, if the contents of I were 3.0 and an indirect recall (RCLi) command were encountered, the contents of R_3 would be recalled. When the content of I is changed, the action of the RCLi is also changed. Because of this relationship, it is possible to access all 26 data storage registers with only one RCLi command.

DSZI (Decrement and Skip on Zero) was designed to help take full advantage of RCLi and other indirect capabilities. A DSZI command causes 1.00 to be subtracted from the contents of I. After the subtraction, the content of I is automatically compared to zero. If the integer part of the value is zero, the calculator skips the step following the DSZI command. If the integer part is non-zero, the following step is executed. This automatic test capability makes DSZI a valuable looping tool.

Steps 102 – 130 of *Tabulator* illustrate a typical use of DSZI and RCLi. The task is to recall the values of the row totals, in order, and output them. Below are the flowchart and the commented code which performs the task.




```
LBL C
CF 1
LBL 6
SPACE     }
RCL I     }
FRAC      }
EEX       }  Store position
4         }  of first register
%         }  location in I
X⇄I       }
DSP 2
LBL 4
RCL i        Indirect recall
F 1?      }  Compute % of grand
GSB E     }  total if flag is set
PRINT X      Output
R↓           Eliminate value from stack
DSZ I        Subtract 1.0 and test
GTO 4        Loop again if I ≠ 0
SPACE     }
RCL 0     }
F 1?      }
GSBE      }  Complete output.
PRINT X   }
R↓        }
STO I     }
R↑
CF 1
RTN
```

NOTES

Tabulator

Step	Key	Comments
001	*LBLa	Clear flag 2 and registers.
002	CF2	
003	CLRG	
004	P⇄S	
005	CLRG	
006	INT	If the value input for number of rows is not in the range of 1 to 24, reject the value.
007	1	
008	X>Y?	
009	GTO2	
010	CLX	
011	2	
012	4	
013	X⇄Y	
014	X≤Y?	
015	GTO0	
016	GTO7	
017	*LBL0	Store # registers + # registers/100 in I.
018	1	
019	%	
020	+	
021	STOI	
022	0	Clear stack.
023	ENT↑	
024	ENT↑	
025	ENT↑	
026	RTN	
027	*LBLA	If flag 2 is set clear stack.
028	F2?	
029	GSB1	
030	ST+i	Add input to row.
031	ST+0	Add input to GT.
032	X⇄Y	Add input to column total.
033	R↓	
034	+	
035	LSTX	Print input?
036	F0?	
037	PRTX	
038	DSZI	Stop if I is not 0.
039	RTN	
040	F0?	Set flag 2 for new stack total.
041	SPC	
042	SF2	
043	RCLI	Reset index for next loop.
044	EEX	
045	4	
046	%	
047	+	
048	STOI	
049	CLX	Print or display column total and stop.
050	ENT↑	
051	R↑	
052	F0?	
053	PRTX	
054	F0?	
055	SPC	
056	RTN	
057	*LBL1	Clear stack except for last input.
058	0	
059	ENT↑	
060	ENT↑	
061	R↑	
062	RTN	
063	*LBLB	If column just changed GTO 1.
064	F2?	
065	GTO1	
066	ISZI	Restore counter. Subtract display from totals.
067	-	
068	LSTX	
069	ST-0	
070	ST-i	
071	F0?	Print space to indicate deletion.
072	SPC	
073	RTN	
074	*LBL1	Reset index to previous column, last value.
075	R↑	
076	RCLI	
077	FRC	
078	1	
079	+	
080	STOI	
081	R↓	Subtract display from totals.
082	-	
083	LSTX	
084	ST-0	
085	ST-i	
086	F0?	Print space to indicate deletion.
087	SPC	
088	RTN	
089	*LBLb	Toggle print/pause flag.
090	F0?	
091	GTO0	
092	SF0	
093	CLX	
094	SPC	
095	1	
096	RTN	
097	*LBL0	
098	CF0	
099	CLX	
100	0	
101	RTN	
102	*LBLC	
103	CF1	Clear % flag.
104	*LBL6	Set index to begin at first row total.
105	SPC	
106	RCLI	
107	FRC	
108	EEX	
109	4	
110	%	
111	X⇄I	
112	DSP2	

REGISTERS

0	1	2	3	4	5	6	7	8	9
GT	used	used	used	used	used	used	used	used	used
S0 used	S1 used	S2 used	S3 used	S4 used	S5 used	S6 used	S7 used	S8 used	S9 used

A	B	C	D	E	I
used	used	used	used	used	index

Program	Comments
113 *LBL4 114 RCLi 115 F1? 116 GSBE 117 PRTX 118 R↓	Recall and output values. If flag 1 is set, convert values to % before output.
119 DSZI 120 GTO4	If I ≠ 0 loop again.
121 SPC 122 RCL0 123 F1? 124 GSBE 125 PRTX	Output grand total or % of grand total if flag 1 is set.
126 R↓ 127 STOI 128 R↑	Return original index to I.
129 CF1 130 RTN	Clear flag 1 and stop.
131 *LBLD 132 SF1 133 GTO6	Output % of total values using LBL C.
134 *LBLE 135 RCL0 136 ÷ 137 EEX 138 2 139 × 140 RTN	Compute % of total for any input value.
141 *LBL2 142 R↓ 143 *LBL7 144 PSE 145 GTO7 146 R/S	Error flash loop.

LABELS					FLAGS
A Val	B Del	C →Tot	D → % Tot	E Val→ % Tot	0 print
a #rows	b P?	c	d	e	1 %
0 used	1 Col Chg	2 error	3	4 Tot	2 Col Chg
5	6 % Tot	7 error	8	9	3

SET STATUS

FLAGS	ON	OFF	TRIG		DISP	
0	☐	☒	DEG	☒	FIX	☒
1	☐	☒	GRAD	☐	SCI	☐
2	☐	☒	RAD	☐	ENG	☐
3	☐	☒			n	2

PRIMARY EXCHANGE SECONDARY REGISTERS

The data storage of your calculator is comprised of 26 registers. Sixteen of these registers are directly accessible at all times through store and recall commands. The remaining 10 secondary registers R_{S0}–R_{S9} are not directly addressable but may be exchanged with primary registers R_0–R_9 at any time. The P⇄S command can be used to do this. Figure 1 represents the action of P⇄S. After execution of the command, the value originally stored in R_{S0} is found in R_0, and the value originally in R_0 is in R_{S0}. A similar exchange would occur between R_1–R_9 and R_{S1}–R_{S9}, respectively.

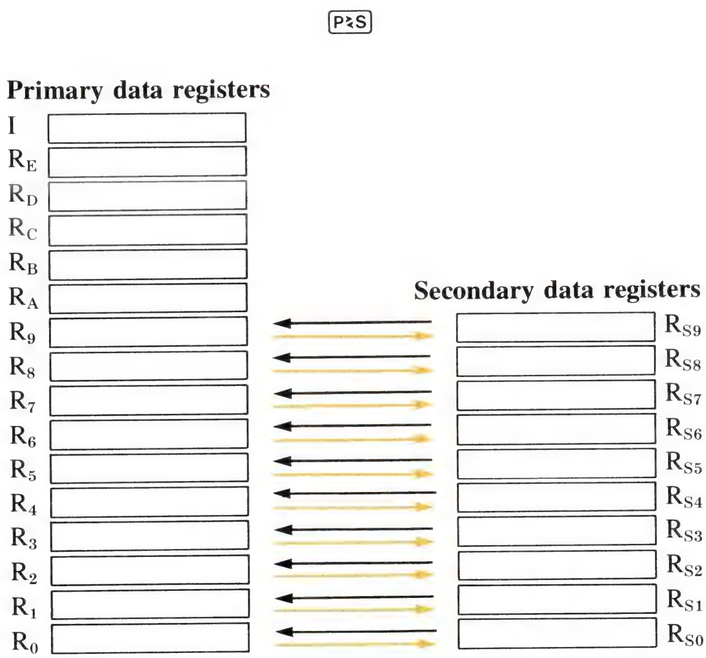


Figure 1.

In *Curve Fitting*, the Σ+ command is used to automatically accumulate the necessary sums in the registers indicated below:

$$\Sigma x \longrightarrow R_{S4}$$
$$\Sigma x^2 \longrightarrow R_{S5}$$
$$\Sigma y \longrightarrow R_{S6}$$
$$\Sigma y^2 \longrightarrow R_{S7}$$
$$\Sigma xy \longrightarrow R_{S8}$$
$$\Sigma n \longrightarrow R_{S9}$$

Before starting to accumulate the sums, registers R_{S4}–R_{S9} must be cleared. Since the clear registers command only operates on the primary registers, a [P⇄S] command is necessary. The code from *Curve Fitting* which prepares the secondary registers for summation is shown below:

[P⇄S]	Exchange primary and secondary registers.
[CL REG]	Clear primary registers.
[P⇄S]	Return cleared registers to secondary status, ready to accumulate sums.

Note that this sequence has no effect on the original, primary registers R_0–R_9. They still contain exactly what they contained before the sequence. This allows R_0–R_9 to be used for user storage during execution of *Curve Fitting*.

After the sums are accumulated, they must be accessed to calculate the regression coefficients a, b and r^2. However, since the sums are in the secondary registers, they are not directly accessible by the store and recall commands. This necessitates use of [P⇄S] again. Label C (steps 68–113) of *Curve Fitting* performs the calculation. [P⇄S] is found at the beginning and the end of the Label C routine. The first [P⇄S] allows the values to be accessed directly. The second [P⇄S] returns the registers to their original configuration.

LBL C	
[P⇄S]	Exchanges primary and secondary registers for access by STO and RCL.
⋮	
[P⇄S]	Exchanges primary and secondary registers returning calculator to original status.
RTN	

Curve Fitting

001	*LBLa	
002	0	Toggle print/pause mode flag.
003	F2?	
004	RTN	
005	1	
006	SF2	
007	RTN	
008	*LBLb	Clear flags and registers for linear regression.
009	CF0	
010	CF1	
011	P⇄S	
012	CLRG	
013	P⇄S	
014	1	
015	RTN	
016	*LBLc	Call LBL b, then set exponential flag.
017	GSBb	
018	SF1	
019	RTN	
020	*LBLd	Call LBL b, then set logarithmic flag.
021	GSBb	
022	SF0	
023	RTN	
024	*LBLe	Call LBL d, then set flag for power curve fit.
025	GSBd	
026	SF1	
027	RTN	
028	*LBLA	Clear Σ– flag.
029	CF3	
030	*LBL8	
031	F2?	Print if flag 2 is set.
032	GSB9	
033	STOD	
034	F1?	ln y if flag 1 set.
035	LN	
036	X⇄Y	ln x if flag 0 is set.
037	STOC	
038	F0?	
039	LN	
040	F3?	If flag 3, then Σ–.
041	GTO0	
042	Σ+	Compute sums.
043	*LBL7	Calculate i + 1.
044	ENT↑	
045	1	
046	+	
047	RCLC	Set inputs in stack positioned for possible deletion.
048	X⇄Y	
049	RCLD	
050	X⇄Y	
051	RTN	
052	*LBL0	Subtract from sums.
053	Σ–	
054	GTO7	
055	*LBL9	Print inputs and reset print flag.
056	SPC	
057	X⇄Y	
058	PRTX	
059	X⇄Y	
060	PRTX	
061	SF2	
062	RTN	
063	*LBLE	
064	SF3	Set Σ– flag.
065	F2?	Print delete indicator if flag is set.
066	GSB9	
067	GTO0	Delete inputs.
068	*LBLC	Switch to secondary registers.
069	P⇄S	
070	SPC	Compute b.
071	RCL8	
072	RCL4	
073	RCL6	
074	×	
075	RCL9	
076	÷	
077	–	
078	ENT↑	
079	ENT↑	
080	RCL4	
081	X²	
082	RCL9	
083	÷	
084	RCL5	
085	X⇄Y	
086	–	
087	÷	
088	STOB	
089	×	Compute r^2.
090	RCL6	
091	X²	
092	RCL9	
093	÷	
094	CHS	
095	RCL7	
096	+	
097	÷	
098	PRTX	
099	RCL6	Compute a.
100	RCL4	
101	RCLB	
102	×	
103	–	
104	RCL9	
105	÷	
106	F1?	
107	e^x	
108	STOA	
109	PRTX	Output a and b.
110	RCLB	
111	PRTX	
112	P⇄S	Switch registers.

REGISTERS

0	1	2	3	4	5	6	7	8	9

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9
0	0	0	0	Σx	Σx^2	Σy	Σy^2	Σxy	n

A	B	C	D	E	I
a	b	x_i	y_i	x,y	0

Step	Key	Comments
113	RTN	
114	*LBLE	Position coefficients in stack for use by projection routines.
115	STOE	
116	RCLA	
117	RCLB	
118	RCLE	
119	F1?	If flag 1 is set, power or exp projection.
120	GTO1	
121	F0?	Logarithmic?
122	LN	
123	×	Linear or logarithmic projection.
124	+	
125	F2?	Print?
126	GTO9	
127	RTN	Stop
128	*LBL1	If flag 0 is set, do power fit.
129	F0?	
130	GTO2	
131	×	Do exponential projection.
132	e^x	
133	×	
134	F2?	Print?
135	GTO9	
136	RTN	Stop
137	*LBL2	Do power projection.
138	X⇄Y	
139	Y^x	
140	×	
141	F2?	Print?
142	GTO9	
143	RTN	Stop.
144	*LBL3	Print −1 indicator.
145	SPC	
146	1	
147	CHS	
148	PRTX	
149	SF2	
150	R↓	
151	RTN	
152	*LBLD	Position coefficients in stack for use by projection routine.
153	STOE	
154	RCLB	
155	1/X	
156	RCLA	
157	RCLE	
158	X⇄Y	
159	F1?	Power or exp?
160	GTO1	
161	−	Linear and log projection.
162	×	
163	F0?	Logarithmic.
164	e^x	
165	F2?	Print?
166	GTO9	
167	RTN	Stop.
168	*LBL1	Power exp calc.
169	÷	
170	F0?	For power GTO 1
171	GTO1	
172	LN	Exponential projection.
173	÷	
174	F2?	Print?
175	GTO9	
176	RTN	Stop
177	*LBL1	Power projection.
178	X⇄Y	
179	Y^x	
180	F2?	Print?
181	GTO9	
182	RTN	Stop.
183	R/S	

LABELS

A $x_i \uparrow y_i$ (+)	B $x_i \uparrow y_i$ (−)	C →r^2, a, b	D $\hat{y} \rightarrow \hat{x}$	E $x \rightarrow \hat{y}$
a P?	b LIN?	c EXP?	d LOG?	e PWR?
0 Σ−	1 used	2 power	3 print	4
5	6	7 display	8 Σ−	9 print

FLAGS

0	Log
1	Exp
2	print
3	Σ−

SET STATUS

FLAGS	ON	OFF	TRIG		DISP	
0	☐	☒	DEG	☒	FIX	☒
1	☐	☒	GRAD	☐	SCI	☐
2	☐	☒	RAD	☐	ENG	☐
3	☐	☒			n	2

MULTIPLE STORAGE IN REGISTERS

In *Calendar Functions* the date is input in mm.ddyyyy format. This allows three pieces of information (the day, the month, and the year) to be carried in one register. In *Calendar Functions* this provides a convenient means of displaying the date. In other programs a similar technique could be used to store more than 26 values in the 26 addressable registers.

When multiple storage techniques are used, two types of code are usually required. The first type breaks a combined number into its individual components. The second type assembles the individual components into a single number.

Steps 83 through 97 of *Calendar Functions* break the date into its individual components.

PROGRAM STEPS	X REGISTER CONTENT	
ENT↑	mm.ddyyyy	(combined form)
INT	mm.000000	
STO7	mm.000000	(months)
—	.ddyyyy	
EEX		
2	100.000000	
X	dd.yyyy00	
ENT↑	dd.yyyy00	
INT	dd.000000	
STO8	dd.000000	(days)
—	.yyyy00	
EEX		
4	10000.000000	
X	yyyy.000000	
STO9	yyyy.000000	(years)

Steps 54 through 78 of *Calendar Functions* assemble the three values into one number for display. However, other operations are being performed which obscure the technique being used. Below is a sample program which could be used to build a date in mm.ddyyyy format if m were stored in R_7, d in R_8, and y in R_9.

PROGRAM STEPS	X REGISTER CONTENTS
RCL7	mm.000000
RCL8	dd.000000
EEX	
2	100.000000
÷	0.dd0000
+	mm.dd0000
RCL9	yyyy.000000
EEX	
6	1000000.000000
÷	0.00yyyy
+	mm.ddyyyy

Calendar Functions

Step	Key	Comments
001	*LBLA	Calculate Δ days and put control 3 in display.
002	RCL4	
003	RCLC	
004	-	
005	3	
006	GTO0	
007	*LBLB	Calculate Δ days and put control 4 in display.
008	RCL3	
009	RCLC	
010	+	
011	4	
012	*LBL0	Store control code.
013	STOI	
014	R↓	Store constants.
015	3	
016	6	
017	5	
018	.	
019	2	
020	5	
021	STO5	
022	3	
023	0	
024	.	
025	6	
026	0	
027	0	
028	1	
029	STO6	
030	R↓	Return Δ days to display.
031	R↓	
032	F3?	If data input, GTO 1.
033	GTO1	
034	STO(i)	Store Δ days according to control code.
035	1	
036	2	
037	2	Calculate y′.
038	.	
039	1	
040	-	
041	RCL5	
042	÷	
043	INT	
044	STO9	Calculate m′.
045	RCL5	
046	×	
047	INT	
048	RCL(i)	
049	-	
050	CHS	
051	STOA	
052	RCL6	
053	÷	
054	INT	
055	STO7	Calculate day of month.
056	RCLA	
057	X⇄Y	
058	RCL6	
059	×	
060	INT	
061	-	
062	STO8	
063	RCL7	Build (m′ − 1). dd part of display.
064	1	
065	RCL8	
066	%	
067	-	
068	-	
069	RCL7	Correct m′ − 1 and y′ to m and y.
070	1	
071	4	
072	÷	
073	GSB2	
074	RCL9	Finish building mm.ddyyyy result and display final answer.
075	EEX	
076	6	
077	÷	
078	+	
079	DSP6	
080	RTN	
081	*LBL1	Break date input into the individual components of mm, dd, yyyy.
082	R↓	
083	ENT↑	
084	INT	
085	STO7	
086	-	
087	EEX	
088	2	
089	×	
090	ENT↑	
091	INT	
092	STO8	
093	-	
094	EEX	
095	4	
096	×	
097	STO9	
098	RCL7	m + 1
099	1	
100	+	
101	ENT↑	m + 1 → m′
102	1/X	
103	.	
104	7	y → y′
105	+	
106	CHS	
107	GSB2	
108	RCL6	Compute day number.
109	×	
110	INT	
111	RCL9	
112	RCL5	

REGISTERS

0	1	2	3	4	5	6	7	8	9
			Day #1	Day #2	365.25	30.6001	m	d	y

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9

A	B	C	D	E	I
used		Δ days			control

```
×
INT
+
RCL8
+
STOi
```

Compute Julian day number for output.

```
1
7
2
0
9
8
2
+
DSP0
RTN
```

If input to this routine has absolute value 1 or greater:
y = y ± 1
m = m ± 12

(+ for plus input)

```
*LBL2
INT
ST+9
1
2
×
-
RTN
```

Store input.

```
*LBLC
DSP0
STOC
```

If input flag, stop.

```
F3?
RTN
```

Calculate Δdays and stop.

```
RCL4
RCL3
-
STOC
RTN
```

If input GTO 4.

```
*LBLD
F3?
GTO4
```

Compute Δdays.

```
GSBC
DSP1
```

Convert to Δ weeks.days format.

```
*LBL3
7
÷
INT
LSTX
FRC
.
7
×
+
RTN
```

Convert Δ weeks.days to days and store.

```
*LBL4
DSP0
ENT↑
INT
7
×
X⇄Y
FRC
1
0
×
+
STOC
RTN
```

Calculate day number.

```
*LBLE
SF3
RCL5
5
GSB0
```

Change day number to modulo 7 number.

```
RCLi
5
+
GSB3
LSTX
1
0
×
RTN
R/S
```

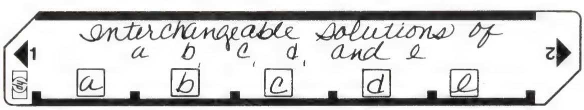
LABELS					FLAGS
A ↔DT$_1$	B ↔DT$_2$	C ↔ΔDays	D ↔ΔWks. Days	E DT → DOW	0
a	b	c	d	e	1
0 calc	1 DT → days	2 m − 12	3 mod 7	4 Δwk → Δday	2
5	6	7	8	9	3 input

SET STATUS					
FLAGS	ON	OFF	TRIG		DISP
0	☐	☒	DEG ☒		FIX ☒
1	☐	☒	GRAD ☐		SCI ☐
2	☐	☒	RAD ☐		ENG ☐
3	☐	☒			n 2

INTERCHANGEABLE SOLUTIONS

In programs like *Annuities and Compound Amounts,* it is necessary to be able to calculate any value given the other values. While there are many ways to do these interchangeable solutions, two methods are designed into your calculator. The method used in *Annuities and Compound Amounts* takes advantage of the STO A through STO E commands. The other method, used in *Calendar Functions,* takes advantage of the keyboard sensing flag (flag 3).

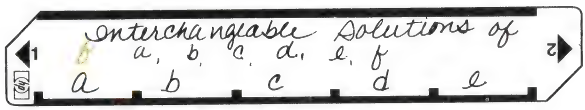
An interchangeable solution requires a method for storage and calculation. It is also desirable to associate inputs and outputs with the mnemonics on the magnetic cards. The STO A through STO E commands accommodate the storage of up to five values in the A through E registers and associate these values with the user definable keys which can be used to initiate calculation. Below is a diagram representing these relationships.



A	B	C	D	E
LBL A	LBL B	LBL C	LBL D	LBL E
C	C	C	C	C
A	A	A	A	A
L	L	L	L	L
C	C	C	C	C
U	U	U	U	U
L	L	L	L	L
A	A	A	A	A
T	T	T	T	T
E	E	E	E	E
a	b	c	d	e
STO A	STO B	STO C	STO D	STO E
RTN	RTN	RTN	RTN	RTN

To store a, press **STO** **A**; to calculate a, press **A**. Note that after any value is calculated, it is automatically stored just before the RTN command stops execution. This eliminates the need to reinput calculated values in subsequent calculations.

The keyboard sensing flag allows up to ten variables to be interchangeably input. It also allows more versatility in storage register selection and allows input processing of data. However, it is slightly more complicated, requires extra steps and may seem mysterious to the uninitiated program user. The diagram below shows the relationships between the magnetic card and the keyboard sensing code.



	A	B	C	D	E
LBL f A	LBL A	LBL B	LBL C	LBL D	LBL E
STO 0	STO 1	STO 2	STO3	STO 4	STO 5
F3?	F3?	F3?	F3?	F3?	F3?
RTN	RTN	RTN	RTN	RTN	RTN
C	C	C	C	C	C
A	A	A	A	A	A
L	L	L	L	L	L
C	C	C	C	C	C
U	U	U	U	U	U
L	L	L	L	L	L
A	A	A	A	A	A
T	T	T	T	T	T
E	E	E	E	E	E
f	a	b	c	d	e
STO 0	STO 1	STO 2	STO 3	STO 4	STO 5
RTN	RTN	RTN	RTN	RTN	RTN

To input the value a, key it in and press **A**. To calculate a, press **A**. Pressing **A** for both input and output works because Flag 3 is set when the digit entry keys are pressed. When Flag 3 is set, the value is stored and execution stops at the first RTN. If the flag is not set (no digit keys were pressed), the program skips the first return and continues through the calculate portion of the program.

Annuities and Compound Amounts

001	*LBLA	Store dummy 0 for n.
002	0	
003	STOA	
004	GSB0	Calculate subroutine.
005	RCLE	Solve for n and store it in R_A.
006	LSTX	
007	-	
008	RCLD	
009	LSTX	
010	-	
011	÷	
012	LN	
013	RCL7	
014	LN	
015	÷	
016	STOA	
017	RTN	
018	*LBLC	Store dummy 1 for PMT.
019	1	
020	STOC	
021	GSB0	Calculate subroutine.
022	1/X	Solve for PMT and store it in R_C.
023	RCLD	
024	R↑	
025	-	
026	×	
027	STOC	
028	RTN	
029	*LBLD	Store dummy 1 for PV.
030	1	
031	STOD	
032	GSB0	Calculate subroutine.
033	+	Solve for PV and store in R_D.
034	STOD	
035	RTN	
036	*LBLE	Calculate subroutine.
037	GSB0	
038	RCLD	Solve for FV (or BAL) and store in R_E.
039	X⇄Y	
040	-	
041	RCL8	
042	÷	
043	STOE	
044	RTN	
045	*LBL0	Clear FV flag.
046	CF1	
047	RCLD	If PV = 0 set FV flag.
048	X=0?	
049	SF1	
050	1	Set annuity due mode off (R_5 = 1).
051	STO5	
052	RCLB	
053	%	Convert i to decimal and store in R_9.
054	STO9	
055	+	Calculate (i + 1).
056	F0?	If AD flag is set store
057	STO5	(1 + i) in R_5.
058	STO7	Store (1 + i) in R_7.
059	RCLA	Calculate $(1 + i)^{-n}$ and store in R_8.
060	CHS	
061	Y^x	
062	STO8	
063	RCLE	FV $(1 + i)^{-n}$
064	×	
065	1	Calculate $[1 - (1 + i)^{-n}]$ and store in R_4.
066	RCL8	
067	-	
068	STO4	Calculate ± (PMT/i). Use – if FV flag is set. Store in R_3.
069	RCLC	
070	RCL9	
071	÷	
072	F1?	
073	CHS	
074	STO3	
075	RCL5	Calculate $\frac{+PMT}{i}[1 - (1 + i)^{-n}]\ R_5$.
076	×	
077	×	
078	RTN	
079	*LBLa	Start by clearing PMT, PV, FV (BAL) registers and annuity due flag.
080	CLX	
081	STOC	
082	STOD	
083	STOE	
084	CF0	
085	RTN	
086	*LBLb	Annuity due flag toggle.
087	F0?	
088	GTO1	
089	1	
090	SF0	
091	RTN	
092	*LBL1	
093	0	
094	CF0	
095	RTN	
096	*LBLB	Clear R_B for sum of i terms.
097	0	
098	STOB	Store address of R_B in R_I for indirect access.
099	2	
100	1	
101	STOI	
102	RCLE	Recall FV, n, and PMT.
103	RCLA	
104	RCLC	If PMT = 0, GTO n, i, PV, FV solution.
105	X=0?	
106	GTO8	Start guess of i. n PMT + BAL.
107	×	
108	+	
109	RCLD	If PV = 0, GTO FV guess.
110	X=0?	
111	GTO3	
112	-	PV guess for i.

REGISTERS

0	1	2	3	4	5	6	7	8	9
			±PMT/i	$[1-(1+i)^{-n}]$	1 or 1 + i	$n(1+i)^{-n-1}$	(1 + i)	$(1 + i)^{-n}$	i/100
S0	S1	S2	S3	S4	S5	S6	S7	S8	S9

A	B	C	D	E	I
n	i	PMT	PV	FV (BAL)	21

Step	Key	Comments
113	RCLA	$\frac{nPMT + BAL - PV}{n}$ and recall PV.
114	÷	
115	RCLD	
116	GTO4	
117	*LBL3	FV guess for i numerator: 2(FV − nPMT)
118	RCLE	
119	LSTX	
120	-	
121	ENT↑	
122	+	and denominator: $(n-1)^2$ PMT + FV
123	RCLA	
124	1	
125	-	
126	X²	
127	RCLC	
128	×	
129	RCLE	
130	+	
131	*LBL4	Guess for i.
132	÷	
133	.	If guess is less than −0.9 use −0.9 for guess.
134	9	
135	CHS	
136	X≤Y?	
137	X⇄Y	
138	GSB5	Store guess as a %.
139	X=0?	If guess = 0 stop.
140	RTN	
141	*LBL6	Calculate f(i).
142	GSB0	
143	+	
144	F1?	
145	CHS	
146	RCLD	
147	-	
148	RCL8	Calculate f′(i).
149	RCLA	
150	RCL7	
151	÷	
152	×	
153	F1?	
154	CLX	
155	STO6	
156	F1?	
157	R↓	
158	F1?	
159	LSTX	
160	RCL4	
161	RCL9	
162	÷	
163	-	
164	RCL5	
165	×	
166	F0?	
167	RCL4	
168	F0?	
169	+	
170	RCLC	
171	×	
172	RCL9	
173	÷	
174	RCL6	
175	RCLE	
176	×	
177	-	
178	÷	f(i)/f′(i)
179	CHS	
180	GSB5	Subtract f(i)/f′(i) from current i value.
181	RCLB	
182	÷	
183	RND	If value is not = to zero, loop again.
184	X≠0?	
185	GTO6	
186	RCLB	Stop and display.
187	RTN	
188	*LBL8	Compute i for n, i, PV, FV problem.
189	RCLE	
190	RCLD	
191	÷	
192	RCLA	
193	1/X	
194	Yˣ	
195	1	
196	-	
197	*LBL5	Convert i to % and add to content of R_B.
198	EEX	
199	2	
200	×	
201	ST+i	
202	RTN	
203	*LBLc	Output n, i, PMT, PV and FV or BAL.
204	SPC	
205	RCLA	
206	PRTX	
207	RCLB	
208	PRTX	
209	RCLC	
210	PRTX	
211	RCLD	
212	PRTX	
213	RCLE	
214	PRTX	
215	RTN	
216	R/S	

LABELS

A n	B i	C PMT	D PV	E FV (BAL)
a start	b AD	c print	d	e
0 calc	1 AD	2	3 FV guess	4 guess
5 i → %	6 loop	7	8 FV,PV-i	9

FLAGS

0	AD
1	PV = 0
2	
3	

SET STATUS

FLAGS	ON	OFF	TRIG		DISP	
0	☐	☒	DEG	☒	FIX	☒
1	☐	☒	GRAD	☐	SCI	☐
2	☐	☒	RAD	☐	ENG	☐
3	☐	☒			n 2	

INDIRECT GTO

The GTO function is used to cause program execution to transfer from the location of the GTO to the label specified. The label may be specified in one of two ways:

1. As a direct branch such as GTO 1, GTO A, GTO f C, etc.
2. As an indirect branch GTOi which causes execution to transfer to the label specified by the content of the I register.

In *Follow Me* the content of the I register is used to specify the operation to be performed. The operation codes are:

CODE	OPERATION
1	+
2	−
3	×
4	÷
5	%
6	I/O HALT
7	Constant

The first time a problem is done using *Follow Me* these codes are stored starting in R_D and ending in R_1. The calculator accesses these codes in subsequent calculations and performs the operations indicated by them.

The GTOi instruction at step 083 actually selects the next operation. The RCLi and X⇆I commands directly above the GTOi place the operation code in the I register. The GTOi command transfers control to one of seven labels corresponding to the operation code stored in the I register. For instance, if 3 is stored in I, the GTOi command will transfer control to LBL3 and the multiply at step 108 will be performed.

NOTES

Follow Me

```
*LBLA
CLRG
P⇄S
CLRG
2
4
STOI
CLX
RTN
```

Clear registers and set index at 24 to begin sequence.

```
*LBLa
+
1
GTO0
```

Perform addition and put addition code of 1 in display register.

```
*LBLb
-
2
GTO0
```

Perform subtraction and put 2 in display, then transfer to LBL 0.

```
*LBLc
×
3
GTO0
```

Perform multiplication and put 3 in display.

```
*LBLd
÷
4
```

Perform division and put 4 in the display.

```
*LBL0
DSZI
```

Decrement step count.

```
GTO1
```

GTO function store.

```
GTO9
```

GTO error.

```
*LBL1
STO(i)
R↓
RTN
```

Store function code and return operation result.

```
*LBLe
%
STOE
CLX
5
GTO0
```

Perform %, store display register value, and put 5 code in display.

```
*LBLB
STOE
CLX
6
GTO0
```

I/0 halt code of 6 put in display after storing display register value.

```
*LBLC
STOE
CLX
7
```

Constant code of 7 put in display after display value is stored.

```
DSZI
GTO1
```

If I is non zero after decrement, store code.

```
*LBL9
CLX
2
4
PSE
GTO9
```

Flash 24 indicating that too many operations have been attempted.

```
*LBL1
STO(i)
CLX
RCLE
```

Store constant code and recall constant value.

```
*LBL8
DSZI
GTO1
```

If I is non zero after dec store cd.

```
GTO9
```

GTO error.

```
*LBL1
STO(i)
CLX
RCLE
RTN
```

Store code and return display to proper status.

```
*LBLD
CLX
2
4
STOI
CLX
STO0
RTN
```

Store 24 in I to reset counter and store zero code in R_0 for auto reset at end of sequence.

```
*LBLE
STOE
R↓
DSZI
RCL(i)
X⇄I
GTO(i)
```

Store display value, access code after dec, put code in I, transfer to LBL corresponding to code.

```
*LBL6
CLX
2
4
STOI
CLX
RCLE
RTN
```

Reset to start new sequence by setting I to 24 and returning output to display.

```
*LBL1
X⇄I
CLX
RCLE
+
GTOE
```

Perform addition and return to LBL E for next instruction.

```
*LBL2
X⇄I
CLX
RCLE
-
GTOE
```

Perform subtraction.

```
*LBL3
X⇄I
CLX
RCLE
×
GTOE
```

Perform multiplication.

```
*LBL4
X⇄I
CLX
```

Perform division.

REGISTERS

0	1	2	3	4	5	6	7	8	9
0	used	used	used	used	used	used	used	used	used

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9
used	used	used	used	used	used	used	used	used	used

A	B	C	D	E	I
used	used	used	used	temp store	step count

113 RCLE 114 ÷ 115 GTOE			
116 *LBL5 117 X⇄I 118 CLX 119 RCLE 120 % 121 GTOE	Perform %.		
122 *LBL6 123 X⇄I 124 CLX 125 RCLE 126 RTN	Halt for I/0.		
127 *LBL7 128 X⇄I 129 CLX 130 RCLE 131 DSZI 132 RCLi 133 GTOE 134 R/S	Recall constant.		

LABELS					FLAGS	SET STATUS		
A Start	B I/O	C Const	D End	E Follow	0	FLAGS	TRIG	DISP
a +	b −	c x	d ÷	e %	1	ON OFF 0 ☐ ☒ 1 ☐ ☒ 2 ☐ ☒ 3 ☐ ☒	DEG ☒ GRAD ☐ RAD ☐	FIX ☒ SCI ☐ ENG ☐ n 2
0 used	1 +	2 −	3 x	4 ÷	2			
5 %	6 I/O	7 const	8	9 error	3			

VARIABLE INPUT

In many instances, it is desirable to input more than one value per user definable key. In *Triangle Solutions,* the lengths of all three sides of a triangle are input with one press of **A**. Before **A** is pressed the values of S_1, S_2, and S_3 must be keyed into the operational stack. The sequence to do this is:

S_1 **ENTER↑** S_2 **ENTER↑** S_3

After this sequence is completed, the operational stack contains the values in the following positions:

T: Unknown value
Z: S_1
Y: S_2
X: S_3

The X, or display register, shows S_3.

To operate successfully, *Triangle Solutions* must store S_1 in R_9, S_2 in R_B and S_3 in R_D. Since S_3 is in the X-register, it can be stored in R_D with a **STO** **D** command (step 002). The value of S_2 must now be moved to the X-register so that they can be stored. A **R↓** function (step 003) is used for this purpose. It moves the Y value to X, the Z value to Y, the T value to Z and the X value to T. After the **R↓**, **STO** **B** is performed placing S_2 in R_B. The operational stack is left as follows:

T: S_3
Z: Unknown value
Y: S_1
X: S_2

Both S_3 and S_2 are stored in the correct registers. After **R↓** and **STO** [9], S_1 is correctly stored. The final stack contents are as follows:

T: S_2
Z: S_3
Y: Unknown value
X: S_1

The complete input sequence is:

LBL A
STO D (store S_3)
R↓
STO B (store S_2)
R↓
STO 9 (store S_1)

Up to four values may be input per user definable key using this type of technique.

Triangle Solutions

001	*LBLA	Store lengths of sides S_3, S_2, S_1.
002	STOD	
003	R↓	
004	STOB	
005	R↓	
006	STO9	
007	R↓	
008	R↓	$P=(S_1+S_2+S_3)/2$
009	+	
010	+	
011	2	
012	÷	
013	STO7	$A_3=2\cos^{-1}\sqrt{\frac{P(P-S_2)}{S_1S_3}}$
014	X²	
015	LSTX	
016	RCLB	
017	x	
018	−	
019	RCL9	
020	RCLD	
021	x	
022	÷	
023	√X	
024	COS⁻¹	
025	2	
026	x	
027	STOE	
028	SIN	$h=S_1 \sin A_3$
029	RCL9	
030	x	
031	STO8	
032	RCL7	$A_2=2\cos^{-1}\sqrt{\frac{P(P-S_1)}{S_2S_3}}$
033	X²	
034	LSTX	
035	RCL9	
036	x	
037	−	
038	RCLB	
039	÷	
040	RCLD	
041	÷	
042	√X	
043	COS⁻¹	
044	2	
045	x	
046	STOC	
047	RCLE	GSB third angle routine.
048	GSB0	
049	STOA	GTO print.
050	GTO1	
051	*LBLB	Store A_1, S_1, and A_3.
052	STOA	
053	R↓	
054	STO9	
055	R↓	
056	STOE	
057	RCLA	GSB third angle
058	GSB0	
059	STOC	$Y = S_1 \sin A_3$.
060	RCLE	
061	RCL9	$X = S_1 \cos A_3$.
062	→R	
063	X⇄Y	
064	STO8	$h = X$.
065	RCLC	$Y = \sin A_2$.
066	1	$X = \cos A_2$.
067	→R	
068	R↓	$S_2=S_1 \sin A_3/\sin A_2$.
069	÷	
070	STOB	
071	R↑	$S_3=S_1 \cos A_3 + S_2 \cos A_2$.
072	x	
073	+	
074	STOD	
075	GTO1	GTO print.
076	*LBLC	Store A_2, A_1, and S_1.
077	STOC	
078	R↓	
079	STOA	
080	R↓	
081	STO9	
082	RCLC	GSB third angle routine.
083	RCLA	
084	GSB0	
085	RCL9	Set stack for A_3, S_1, A_1 solution.
086	RCLA	
087	GTOB	
088	*LBLD	Store S_2, A_1, and S_1.
089	STOB	
090	R↓	
091	STOA	
092	R↓	
093	STO9	
094	RCLA	$S_3{}^2 = S_1{}^2 + S_2{}^2 - 2S_1S_2 \cos A_1$.
095	RCLB	
096	→R	
097	RCL9	
098	−	
099	→P	
100	STOD	
101	RCL9	Recall S_1, S_2, and S_3 and GTO A.
102	RCLB	
103	RCLD	
104	GTOA	
105	*LBLE	Store A_2, S_2, and S_1.
106	STOC	
107	R↓	
108	STOB	
109	R↓	
110	STO9	
111	RCLC	
112	SIN	

REGISTERS

0	1	2	3	4	5	6	7	8	9
							used	h	S_1

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9

A	B	C	D	E	I
A_1	S_2	A_2	S_3	A_3	

Step	Key	Comments
113	RCLB	$A_3 = \sin^{-1}\left(\frac{S_2}{S_1} \sin A_2\right)$
114	×	
115	RCL9	
116	÷	
117	SIN⁻¹	
118	STOE	
119	RCLC	GSB third angle.
120	GSB0	
121	STOA	Recall A_3, S_1, and A_1 and GSB B.
122	RCLE	
123	RCL9	
124	RCLA	
125	GSBB	
126	RCL9	Stop if this is the only solution.
127	RCLB	
128	X≤Y?	
129	GTO9	
130	RCLE	Find secondary angle for alternate solution.
131	COS	
132	CHS	
133	COS⁻¹	
134	STOE	
135	RCLC	GSB third angle.
136	GSB0	
137	STOA	Recall A_3, S_1, and A_1 and GSB B.
138	RCLE	
139	RCL9	
140	RCLA	
141	GTOB	
142	*LBL0	Third angle = $\cos^{-1}[-\cos(A+B)]$
143	+	
144	COS	
145	CHS	
146	COS⁻¹	
147	RTN	
148	*LBL1	
149	SPC	Print values starting with S_1.
150	SPC	
151	RCL9	
152	PRTX	
153	RCLA	
154	PRTX	
155	SPC	
156	RCLB	
157	PRTX	
158	RCLC	
159	PRTX	
160	SPC	
161	RCLD	
162	PRTX	
163	RCLE	
164	PRTX	
165	SPC	
166	RCL8	Calculate and print area = $(S_1\ S_3 \sin A_3)/2$.
167	RCLD	
168	×	
169	2	
170	÷	
171	PRTX	
172	RTN	
173	*LBL9	Area.
174	R↓	
175	R↓	
176	RTN	
177	R/S	

LABELS					FLAGS
A S_1, S_2, S_3	B A_3, S_1, A_1	C S_1, A_1, A_2	D S_1, A_1, S_2	E S_1, S_2, A_2	0
a	b	c	d	e	1
0 3rd angle	1 print	2	3	4	2
5	6	7	8	9 Area	3

SET STATUS				
FLAGS	ON	OFF	TRIG	DISP
0	☐	☒	DEG ☒	FIX ☒
1	☐	☒	GRAD ☐	SCI ☐
2	☐	☒	RAD ☐	ENG ☐
3	☐	☒		n 2

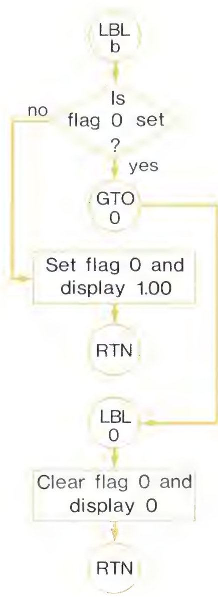
FLAG SET, CLEAR AND TEST—COMMAND CLEARING FLAGS

Review of the input values for *Vector Operations* is an option available to the user. When the program is loaded, the non-review status is automatically set. The user can change this status by pressing f B. Each time the f B keys are pressed, the status is changed and 1.00 or 0.00 is displayed to indicate whether or not the input values will be reviewed. The 1.00 indicates review and the 0.00 indicates no review.

Flag 0 and flag 1 are command clearing flags. That is, once they are set they remain set until a clear flag command is encountered. Testing them has no effect on their on/off status.

Flag 0 is used to control the review of the input values in *Vector Operations*. Lines 064, 090 and 112 contain PRST (print stack).* Preceding each of these statements is F0? (test flag 0). If flag 0 is set the PRST commands will be executed, reviewing the input values. If flag 0 is not on, the PRST commands are skipped. Below is the code used to change the flag status.

If flag 0 is off, this code sets flag 0 on and displays 1.00. If flag 0 is on, this code turns flag 0 off and displays 0.00.




```
LBL b
F0?
GTO 0
SF0
1
RTN
LBL 0
CF0
0
RTN
```

*The HP-67 interprets PRST as pause stack. The values contained in the T, Z, Y, and X registers will be displayed for approximately 3 seconds each. The decimal point will flash, indicating program execution will resume automatically.

NOTES

Vector Operations

Step	Key	Comment
001	*LBLa	
002	F1?	Toggle two and three dimensional modes.
003	GTO0	
004	SF1	
005	3	
006	RTN	
007	*LBL0	
008	2	
009	CF1	
010	RTN	
011	*LBLb	Toggle print/pause mode.
012	F0?	
013	GTO0	
014	SF0	
015	1	
016	RTN	
017	*LBL0	
018	CF0	
019	0	
020	RTN	
021	*LBLD	Store magnitude and input 1 code.
022	STO7	
023	1	
024	GTO0	
025	*LBLE	Store magnitude and input 2 code.
026	STO8	
027	2	
028	*LBL0	
029	SF2	GSB S→C routine.
030	GSB5	
031	GTOi	GTO storage routine.
032	*LBL1	Storage for vector 1.
033	STO9	
034	R↓	
035	STOA	
036	R↓	
037	STOB	
038	1	
039	RTN	
040	*LBL2	Storage for vector 2.
041	STOC	
042	R↓	
043	STOD	
044	R↓	
045	STOE	
046	2	
047	RTN	
048	*LBLd	Keyboard S→C begins.
049	0	
050	*LBL5	
051	STOI	Store code.
052	R↑	
053	F1?	
054	GTO0	If 3D mode is set, skip $\pi/2$ substitution for Z register
055	CLX	
056	1	
057	SIN-1	content.
058	*LBL0	
059	R↓	Put vector code in T.
060	CLX	
061	RCLI	
062	R↓	
063	F0?	Print input?
064	PRST	
065	X⇄Y	Convert S→C.
066	1	
067	→R	
068	R↑	
069	R↑	
070	→R	
071	X⇄Y	
072	R↑	
073	X⇄Y	
074	×	
075	LSTX	
076	R↑	
077	×	
078	GTO2	Begin C→S.
079	*LBLe	If 2D, set content of Z register to zero.
080	R↓	
081	R↓	
082	F1?	
083	GTO0	
084	CLX	
085	*LBL0	
086	R↓	Set T to zero.
087	CLX	
088	R↓	
089	F0?	Print input?
090	PRST	
091	*LBL6	
092	→P	Convert C→S.
093	X⇄Y	
094	X<0?	
095	GSB3	
096	R↓	
097	X⇄Y	
098	F1?	
099	GTO0	
100	CLX	
101	*LBL0	
102	→P	
103	R↑	
104	X⇄Y	
105	*LBL2	
106	R↑	Put zero in T register.
107	CLX	
108	R↓	
109	F2?	Return if GSB.
110	RTN	
111	F0?	Print result?
112	PRST	

REGISTERS

0	1	2	3	4	5	6	7	8	9
							r_1	r_2	x_1
S0	S1	S2	S3	S4	S5	S6	S7	S8	S9

A	B	C	D	E	I
y_1	z_1	x_2	y_2	z_2	code

113	RTN	
114	*LBL3	Convert negative angles to positive angles 0° – 360°.
115	1	
116	CHS	
117	COS⁻¹	
118	+	
119	LSTX	
120	+	
121	RTN	
122	*LBLA	Add $\vec{V}_1$ and $\vec{V}_2$ and convert back to spherical coordinates.
123	RCLB	
124	RCLE	
125	+	
126	RCLD	
127	RCLA	
128	+	
129	RCLC	
130	RCL9	
131	+	
132	SF2	
133	GSB6	
134	PRST	
135	RTN	
136	*LBLB	Take cross product.
137	RCL9	
138	RCLD	
139	×	
140	RCLA	
141	RCLC	
142	×	
143	-	
144	RCLB	
145	RCLC	
146	×	
147	RCL9	
148	RCLE	
149	×	
150	-	
151	RCLB	
152	RCLD	
153	×	
154	STOI	
155	CLX	
156	RCLA	
157	RCLE	
158	×	
159	RCLI	
160	-	
161	→P	Convert back to spherical.
162	X⇄Y	
163	X<0?	
164	GSB3	
165	R↓	
166	X⇄Y	
167	→P	
168	R↑	

169	X⇄Y	
170	R↑	
171	CLX	
172	R↓	
173	PRST	
174	RTN	
175	*LBLC	Take dot product.
176	SPC	
177	RCL7	
178	RCL8	
179	×	
180	1/X	
181	RCL9	
182	RCLC	
183	×	
184	RCLA	
185	RCLD	
186	×	
187	+	
188	RCLB	
189	RCLE	
190	×	
191	+	
192	PRTX	Compute angle between vectors.
193	×	
194	LSTX	
195	X⇄Y	
196	COS⁻¹	
197	PRTX	
198	RTN	
199	R/S	

LABELS					FLAGS
A $\vec{V}_1 + \vec{V}_2$	B $\vec{V}_1 \times \vec{V}_2$	C $\vec{V}_1 \cdot \vec{V}_2$	D $\phi_1 \uparrow \theta_1 \uparrow r_1$	E $\phi_2 \uparrow \theta_2 \uparrow r_2$	0 PRINT?
a 3D/2D?	b P?	c	d S→C	e C→S	1 3D/2D?
0 used	1 $\vec{V}_1$	2 $\vec{V}_2$, print	3 0° – 360°	4	2 S→C
5 S→C	6 C→S	7	8	9	3

SET STATUS				
FLAGS	ON	OFF	TRIG	DISP
0	☐	☒	DEG ☒	FIX ☒
1	☐	☒	GRAD ☐	SCI ☐
2	☐	☒	RAD ☐	ENG ☐
3	☐	☒		n 2

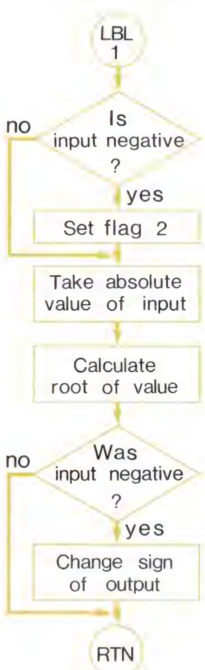
FLAG SET, CLEAR AND TEST-TEST CLEARING FLAG

Flag 2 and flag 3* are test clearing flags. Each time they are tested, they are automatically cleared. This makes them especially useful in many programming situations.

In *Polynomial Evaluation,* flag 2 is used twice. At step 62 it is used to decide whether to add or subtract; and at step 145, it is used to determine whether a result should be positive or negative. The following discussion details the use in the latter case.

Label 1 calculates the cube root of a number. This would be very simple if y^x were defined for the case where y is negative and x is a non-integer. However, if we tried to find the cube root of –8 (which is –2) directly, we would obtain an error message. The following flow chart and code yield the desired result:

Flow chart



Code	X register (Positive 8)	X register (Negative 8)
LBL 1	8	-8
x<0	8	-8
SF2	8	-8
ABS	8	8
3 1/x y^x	3 0.333... 2	3 0.333... 2
F2?	2	2
CHS	2	-2
RTN	2	-2

*When using flag 3, you must be aware that it is set whenever the numeric keys are pressed.

NOTES

Polynomial Evaluation

Step	Key	Comment
001	*LBLa	Store zero for degree, to initialize.
002	0	
003	STOE	
004	RTN	
005	*LBLB	Store a_0 and set degree indicator (= degree + 1) to 1.
006	STO1	
007	1	
008	RTN	
009	*LBLC	Store a_1 and set indicator to 2.
010	STO2	
011	2	
012	GTO0	
013	*LBLD	Store a_2 and set indicator to 3.
014	STO3	
015	3	
016	GTO0	
017	*LBLE	Store a_3 and set indicator to 4.
018	STO4	
019	4	
020	*LBL0	Sort to find and retain largest indicator.
021	X⇄Y	
022	X=0?	
023	RTN	
024	X⇄Y	
025	RCLE	
026	X⇄Y	
027	X>Y?	
028	STOE	
029	X⇄Y	
030	R↓	
031	RTN	
032	*LBLb	Start polynomial solution.
033	SPC	
034	RCLE	
035	STOI	Put degree code in I for control.
036	÷	
037	RCLi	Divide all coef. by coef. of highest deg.
038	STOA	
039	1/X	
040	GSB5	
041	RCL1	Select proper deg solution.
042	CHS	
043	RCL2	
044	GTOi	
045	*LBL3	Begin quadratic equation.
046	RCL1	
047	*LBL9	
048	STOB	Calculate $-\frac{a_1}{2a_2}$
049	X⇄Y	
050	CHS	
051	2	
052	÷	
053	X<0?	Set flag to det sol order.
054	SF2	
055	ENT↑	$(a_1/2a_2)^2-(a_0/a_2)$
056	X²	
057	RCLB	
058	–	
059	X<0?	Imaginary roots?
060	GTO0	
061	√X	
062	F2?	Compute x_1 (the root of largest absolute value).
063	CHS	
064	+	Compute x_2.
065	÷	
066	LSTX	
067	GTO6	
068	*LBL0	Compute imaginary part.
069	ABS	
070	√X	
071	1	Output img code.
072	CHS	
073	PRTX	
074	R↓	Img part to X.
075	*LBL6	Output x_2 or img part.
076	PRTX	
077	*LBL2	Output x_1 or real part.
078	X⇄Y	
079	PRTX	
080	RCLA	
081	*LBL5	Return equation to original form.
082	ST×4	
083	ST×3	
084	ST×2	
085	ST×1	
086	R↓	Stop and display.
087	CF2	
088	RTN	
089	*LBL4	Start 3rd degree solution by computing Q.
090	3	
091	÷	
092	RCL3	
093	X²	
094	9	
095	÷	
096	–	
097	STOD	
098	3	Compute Q^3.
099	Yˣ	
100	STOC	
101	RCL3	Compute R.
102	RCL2	
103	×	
104	RCL1	
105	3	
106	×	
107	–	
108	6	
109	÷	
110	RCL3	
111	3	
112	Yˣ	

REGISTERS

0	1	2	3	4	5	6	7	8	9
	a_0	a_1	a_2	a_3					

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9

A	B	C	D	E	I
a_{high}	R, X, a_0/a_2	Q^3	Q	degree	control

Step	Key	Comments
113	2	
114	7	
115	÷	
116	-	
117	STOB	
118	X²	$Q^3 + R^2$ decision.
119	+	
120	X<0?	
121	GTO0	
122	√X	Compute x_3 using $x_3 = S + T - \frac{a_2}{3a_3}$
123	RCLB	
124	X⇄Y	
125	-	
126	LSTX	
127	RCLB	
128	+	
129	GSB1	
130	X⇄Y	
131	GSB1	
132	+	
133	RCL3	
134	3	
135	÷	
136	-	
137	GTO8	
138	*LBL1	Cube root of a number.
139	X<0?	
140	SF2	
141	ABS	
142	3	
143	1/X	
144	Yˣ	
145	F2?	
146	CHS	
147	RTN	
148	*LBL0	Compute x_3 using $x = 2\sqrt{-Q}\ \cos(M) - \frac{a_2}{3a_3}$ Where $M = \frac{1}{3}\cos^{-1}(R/\sqrt{-Q^3})$
149	RCLB	
150	RCLC	
151	CHS	
152	√X	
153	÷	
154	COS⁻¹	
155	3	
156	÷	
157	COS	
158	RCLD	
159	CHS	
160	√X	
161	×	
162	ENT↑	
163	+	
164	RCL3	
165	3	
166	÷	
167	-	
168	*LBL8	Output x_3 and begin synthetic division.
169	PRTX	
170	SPC	
171	STOB	
172	RCL3	
173	+	
174	ENT↑	
175	ENT↑	
176	RCLB	
177	×	
178	RCL2	
179	+	
180	GTO9	
181	*LBLA	Set up for polynomial evaluation.
182	ENT↑	
183	ENT↑	
184	ENT↑	
185	RCLE	
186	STOI	
187	CLX	
188	RCLi	
189	DSZI	
190	GTOd	Degree one check.
191	RTN	
192	*LBLd	Evaluate f(x).
193	×	
194	RCLi	
195	+	
196	DSZI	
197	GTOd	
198	RTN	Stop and display.
199	R/S	

LABELS					FLAGS
A $x \to f(x)$	B a_0	C a_1	D a_2	E a_3	0
a Start	b →Solve	c	d	e	1
0 used	1 cube root	2 output x_1	3 deg 2	4 deg 3	2 sign
5 divide	6 output x_2	7 used	8 syn div	9 deg 2	3

SET STATUS

FLAGS	ON	OFF	TRIG		DISP	
0	☐	☒	DEG	☒	FIX	☒
1	☐	☒	GRAD	☐	SCI	☐
2	☐	☒	RAD	☐	ENG	☐
3	☐	☒			n	2

SUBROUTINES AND INDIRECT RECALLS

LBL a (lines 22 through 49) of *Matrix Operations* calculates the determinant of the 3 × 3 matrix stored in registers R_1 through R_9.

$$\begin{vmatrix} R_1 & R_2 & R_3 \\ R_4 & R_5 & R_6 \\ R_7 & R_8 & R_9 \end{vmatrix} = (R_5R_9 - R_6R_8)\,R_1 - (R_4R_9 - R_6R_7)\,R_2 + (R_4R_8 - R_5R_7)\,R_3$$

$$= -(R_6R_8R_1 + R_4R_9R_2 + R_5R_7R_3) + R_3R_8R_4 + R_1R_9R_5 + R_2R_7R_6$$

The following keystroke procedure will perform the calculation:

RCL 6 RCL 8 <u>RCL 1</u> × × RCL 4 RCL 9 <u>RCL 2</u> × × + RCL 5
RCL 7 <u>RCL 3</u> × × + CHS RCL 3 RCL 8 <u>RCL 4</u> × × + RCL 1
RCL 9 <u>RCL 5</u> × × + RCL 2 RCL 7 <u>RCL 6</u> × × +

There are two patterns in the above procedure which can be exploited to reduce the number of program steps necessary for solution:

1. × × + appears repeatedly.
2. The values recalled immediately before × × +, are recalled from consecutive registers (note underlined RCL instructions in keystrokes above).

A subroutine can be used to take advantage of item one, while indirect recalls in combination with the ISZ command can be used to recall values consecutively. Let's examine the code that does this.

```
022  *LBLa
023      0
024   STOI
025   RCL6
026   RCL8
027   GSB7
028   RCL4
029   RCL9
030   GSB7
031   RCL5
032   RCL7
033   GSB7
034    CHS
035   RCL3
036   RCL6
037   GSB7
038   RCL1
039   RCL9
040   GSB7
041   RCL2
042   RCL7
043  *LBL7
044   ISZI
045   RCLi
046      ×
047      ×
048      +
049    RTN
```

Here is what happens on the first, second and sixth time the subroutine is executed.

Step				
044	ISZI →	$I = 1$	$I = 2$	$I = 6$
045	RCLi →	RCL 1	RCL 2	RCL 6
046	× →	$R_8 \times R_1$	$R_9 \times R_2$	$R_7 \times R_6$
047	× →	$R_6 \times R_8 \times R_1$	$R_4 \times R_9 \times R_2$	$R_2 \times R_7 \times R_6$
048	+ →	$0 + R_6 \times R_8 \times R_1$	Subtotal	Total
049	RTN →	Return to call	Return to call	Stop

Each time the GSB 7 command is encountered, the calculator goes to LBL 7, executes the ISZ command, which adds one to the contents of register I, and recalls the contents of the register specified by the contents of register I (R_1 through R_6). After this, the × × + is done and execution continues at the step following the GSB 7 call.

Matrix Operations

Step	Key	Comment
001	*LBLA	Set 0 in display for indirect store.
002	0	
003	GTO5	
004	*LBLB	Set 3 in display for indirect store.
005	3	
006	GTO5	
007	*LBLC	Set 6 in display for indirect store.
008	6	
009	GTO5	
010	*LBLD	Set 19 in display for indirect store.
011	1	
012	9	
013	*LBL5	Store code in I.
014	STOI	
015	GSB6	Store three input values in proper registers according to code.
016	GSB6	
017	*LBL6	
018	R↑	
019	ISZI	
020	STOi	
021	RTN	
022	*LBLa	Calculate determinant.
023	0	
024	STOI	
025	RCL6	
026	RCL8	
027	GSB7	
028	RCL4	
029	RCL9	
030	GSB7	
031	RCL5	
032	RCL7	
033	GSB7	
034	CHS	
035	RCL3	
036	RCL8	
037	GSB7	
038	RCL1	
039	RCL9	
040	GSB7	
041	RCL2	
042	RCL7	
043	*LBL7	
044	ISZI	
045	RCLi	
046	×	
047	×	
048	+	
049	RTN	
050	*LBLb	Calculate reciprocal of determinant.
051	GSBa	
052	1/X	
053	RCL1	Calculate inverse.
054	RCL9	
055	×	
056	RCL3	

Step	Key
057	RCL7
058	GSB3
059	STOD
060	CLX
061	RCL3
062	RCL4
063	×
064	RCL1
065	RCL6
066	GSB3
067	STOE
068	CLX
069	RCL2
070	RCL7
071	×
072	RCL1
073	RCL8
074	GSB3
075	STOI
076	CLX
077	RCL1
078	RCL5
079	×
080	RCL2
081	RCL4
082	GSB3
083	STO0
084	CLX
085	RCL3
086	RCL8
087	×
088	RCL2
089	RCL9
090	GSB3
091	STO1
092	CLX
093	RCL2
094	RCL6
095	×
096	RCL3
097	RCL5
098	GSB3
099	STO3
100	CLX
101	RCL5
102	RCL9
103	×
104	RCL6
105	RCL8
106	GSB3
107	STO2
108	CLX
109	RCL6
110	RCL7
111	×
112	RCL4

REGISTERS

0	1	2	3	4	5	6	7	8	9
γ_3	a_1, α_1	a_2, α_2	a_3, α_3	b_1, β_1	b_2, β_2	b_3, β_3	c_1, γ_1	c_2, γ_2	c_3, γ_3
S0	S1	S2	S3	S4	S5	S6	S7	S8	S9

A	B	C	D	E	I
d_1	d_2	d_3	β_2	β_3	control

Step	Key	Comment
113	RCL9	
114	GSB3	
115	STO6	
116	CLX	
117	RCL4	
118	RCL8	
119	×	
120	RCL5	
121	RCL7	
122	GSB3	
123	RCLI	
124	RCL0	Store inverse values in proper registers.
125	GSBC	
126	RCL2	
127	RCL1	
128	RCL3	
129	GSBA	
130	RCL6	
131	RCLD	
132	RCLE	
133	GSBB	
134	CLX	
135	RTN	Stop and display 0.
136	*LBL3	Inverse subroutine.
137	×	
138	-	
139	×	
140	RTN	
141	*LBLE	Initialize output loop.
142	SPC	
143	1	
144	STOI	
145	*LBL2	
146	RCLi	Output registers R_1 through R_9.
147	PRTX	
148	9	
149	RCLI	
150	X=Y?	
151	GTO0	
152	3	
153	÷	
154	FRC	
155	X=0?	
156	SPC	
157	RCLI	
158	ISZI	
159	GTO2	
160	*LBL0	
161	SPC	Output registers R_A through R_C.
162	RCLA	
163	PRTX	
164	RCLB	
165	PRTX	
166	RCLC	
167	PRTX	
168	RTN	
169	*LBLc	
170	SPC	First value from multiplication.
171	1	
172	STOI	
173	GSB1	
174	STOD	Second value from multiplication.
175	2	
176	STOI	
177	GSB1	
178	STOE	
179	3	Third value from multiplication.
180	STOI	
181	GSB1	
182	STO0	
183	0	Put values in stack for display.
184	RCLD	
185	RCLE	
186	RCL0	
187	RTN	
188	*LBL1	Multiplication.
189	0	
190	RCLA	
191	GSB4	
192	RCLB	
193	GSB4	
194	RCLC	
195	GSB4	
196	PRTX	
197	RTN	
198	*LBL4	Multiplication subroutine.
199	RCLi	
200	×	
201	+	
202	ISZI	
203	ISZI	
204	ISZI	
205	RTN	
206	R/S	

LABELS

A a_1, a_2, a_3	B b_1, b_2, b_3	C c_1, c_2, c_3	D d_1, d_2, d_3	E Print
a →Det	b →Inv	c →Mult	d	e
0 print	1 mult	2 print	3 inv	4 mult
5 code	6 input	7 det	8	9

FLAGS

0	
1	
2	
3	

SET STATUS

FLAGS	ON	OFF
0	☐	☒
1	☐	☒
2	☐	☒
3	☐	☒

TRIG	
DEG	☒
GRAD	☐
RAD	☐

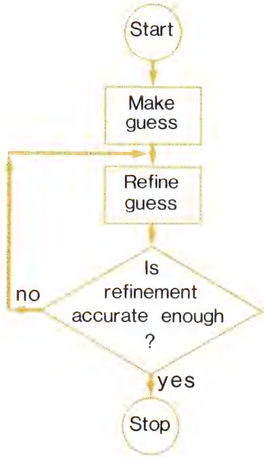
DISP	
FIX	☒
SCI	☐
ENG	☐
n	2

ITERATIVE TEST AND LOOP

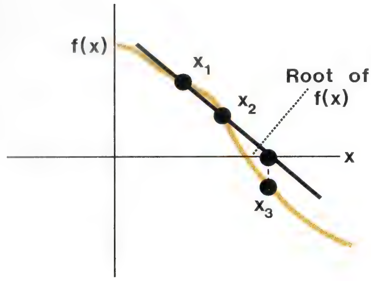
Some equations cannot be solved explicitly. That is, it is impossible to separate a particular variable from the rest of the equation. Solution of this type of equation requires a repetitive technique. In general, such techniques are composed of three basic operations.

1. An initial guess is made.
2. This guess is refined.
3. The refined guess is tested for accuracy. If the accuracy is satisfactory, the result is displayed. If the result is not satisfactory, the refinement process is repeated.

In flow chart form, the process might look like this:



In *Calculus And Roots Of f(x),* LBL E (steps 83 through 112) performs a general interative solution for user-specified functions. The initial guess supplied by the user is refined using the secant method. The secant method evaluates the function f(x) at two points and generates a third refined point. Graphically, this can be represented by the sketch below:



By defining a straight line using x_1 and x_2, x_3 can be found. Subsequently, x_2 and x_3 can be used to generate x_4 etc.

The equation defining the secant method is as follows:

$$x_{i+1} = x_i - f(x_i)\left(\frac{(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}\right)$$

It is evaluated repeatedly by steps 88 through 103. Each time these steps are repeated, the value of x is refined.

Steps 104 through 110 (excluding steps 105 and 106) test to determine whether the guess has been refined to the desired accuracy. If another loop is required, control is transferred to LBL 6. If the value is sufficiently accurate, the program stops, displaying the result at step 112.

The display setting, in combination with the RND function, is used to determine the accuracy of the result. If the amount of change in x_i divided by x_{i+1} rounds to zero, the condition for convergence is met and x_{i+1} is displayed as the answer. If the rounded value is not zero, another iteration is required. For instance, if $x_i = 10$, the change in x_i is 0.1 and the display is set at two decimal places, the test value would be calculated as follows:

$$\begin{aligned}\text{Test value} &= \text{RND}(0.1/(10-0.1)) = \text{RND}(0.01010101)\\ &= 0.01\end{aligned}$$

Since the value is not zero, another loop is required. If, on the next loop, the refinement were 0.01, and x_i were 9.9, the test value would be calculated as follows:

$$\begin{aligned}\text{Test value} &= \text{RND}(0.01/(9.9-0.01)) = \text{RND}(0.001011122)\\ &= 0.00\end{aligned}$$

Since the value is zero, x_{i+1} would be displayed as the result ($x_{i+1} = 9.89$). Note that, if the display had been set to three decimal places, another loop would be required since the RND function is display dependent.

Calculus and Roots of f(x)

Step	Key	Comment
001	*LBLA	Store function number.
002	STOI	
003	RTN	
004	*LBLe	Pause toggle.
005	F0?	
006	GTO0	
007	SF0	
008	1	
009	RTN	
010	*LBL0	
011	0	
012	CF0	
013	RTN	
014	*LBLa	Store %Δ and set flag.
015	SF1	
016	STOE	
017	RTN	
018	*LBLE	Choose default %Δ or use 0.01%?
019	EEX	
020	CHS	
021	2	
022	RCLE	
023	F1?	
024	X⇄Y	
025	R↓	
026	%	If x=0 use %Δ rather than % of x as Δx.
027	X=0?	
028	LSTX	
029	STOC	
030	2	f(x − Δx/2).
031	÷	
032	−	
033	STOA	
034	STO0	
035	GSBi	
036	STOD	
037	RCLA	f(x + Δx/2).
038	RCLC	
039	+	
040	STO0	
041	GSBi	
042	STOB	$\frac{f(x+\Delta x/2)-f(x-\Delta x/2)}{\Delta x}$
043	RCLD	
044	−	
045	RCLC	
046	÷	
047	RTN	
048	*LBLC	f(x).
049	STO0	
050	GSBi	
051	RTN	
052	*LBLD	Store a.
053	X⇄Y	
054	STO0	b−a.
055	−	
056	X⇄Y	Store n.
057	STOB	
058	÷	(b − a)/n
059	STOC	
060	2	$\frac{b-a}{2n}$
061	÷	
062	ST+0	
063	0	Set integral sum at 0.
064	STO9	
065	RCLB	Put number of intervals in I.
066	X⇄I	
067	*LBL7	Return function number to I and n to R_B.
068	X⇄I	
069	STOB	
070	RCL0	$F'(R_0)$
071	GSBi	
072	RCLC	
073	ST+0	R_0 + (b − a)/n
074	×	Add $f(R_0)$ (b − a)/n
075	ST+9	
076	RCLB	Decrement n and save function in display.
077	X⇄I	
078	DSZI	
079	GTO7	Store function number.
080	STOI	
081	RCL9	Display result of integration.
082	RTN	
083	*LBLE	Use numerical differential to generate x_i from user guess.
084	FIX	
085	GSBB	
086	RCLB	
087	GTO0	
088	*LBL6	Evaluate $f(x_i)$
089	RCL0	
090	GSBi	
091	STOB	
092	*LBL0	Secant method calculates correction for x value and sets values for next loop.
093	RCLA	
094	RCL0	
095	STOA	
096	−	
097	RCLD	
098	RCLB	
099	STOD	
100	−	
101	÷	
102	×	
103	ST−0	Subtract correction.
104	RCL0	Pause and display root if flag set?
105	F0?	
106	PSE	
107	÷	RND (change/x_{i+1})
108	RND	
109	X≠0?	Accurate to display?
110	GTO6	
111	RCLA	If it is, display result.
112	RTN	

REGISTERS

0	1	2	3	4	5	6	7	8	9
x									integral
S0	S1	S2	S3	S4	S5	S6	S7	S8	S9

A	B	C	D	E	I
x_{i-1}	$f(x_i)$	Δx	$f(x_{i-1})$	%Δ	function

Step	Key	Comments
001	*LBL1	Graphical evaluation subroutine.
002	R/S	
003	RTN	
004	*LBL2	$f(x) = \tan(x) - \text{Inv}(x) - x$
005	RAD	
006	TAN	
007	LSTX	
008	-	
009	RCL2	
010	-	
011	DEG	
012	RTN	
013	*LBL3	$f(\theta) = \dfrac{1}{\sqrt{1 - k^2 \sin^2 \theta}}$
014	RAD	
015	SIN	
016	RCL1	
017	x	
018	X²	
019	1	
020	X⇄Y	
021	-	
022	√X	
023	1/X	
024	DEG	
025	RTN	

LABELS

A Function #	B x→f'(x)	C x→f(x)	D n↑a↑b→∫	E x_0→root
a %Δ	b	c	d	e pause
0 used	1	2	3	4
5	6 iterate	7 integrate	8	9

FLAGS

0	Pause
1	%Δ
2	
3	

SET STATUS

FLAGS	ON	OFF
0	☐	☒
1	☐	☒
2	☐	☒
3	☐	☒

TRIG	
DEG	☒
GRAD	☐
RAD	☐

DISP	
FIX	☒
SCI	☐
ENG	☐
n	2

English—SI Conversions (Metric Conversions)

Step	Key	Comment
001	*LBLa	Set millimeter inch flag.
002	SF2	
003	*LBLA	
004	2	Input conversion constant.
005	5	
006	.	
007	4	
008	F2?	in. to mm or mm to in?
009	1/X	
010	X⇄Y	Set stack for LST x
011	x	
012	RTN	Convert.
013	*LBLb	Feet-meter conversion.
014	SF2	
015	*LBLB	
016	.	
017	3	
018	0	
019	4	
020	8	
021	F2?	
022	1/X	
023	X⇄Y	
024	x	
025	RTN	
026	*LBLc	Gallon-liter conversion.
027	SF2	
028	*LBLC	
029	3	
030	.	
031	7	
032	8	
033	5	
034	4	
035	1	
036	1	
037	7	
038	8	
039	4	
040	F2?	
041	1/X	
042	X⇄Y	
043	x	
044	RTN	
045	*LBLd	Pound force-newton conversion.
046	SF2	
047	*LBLD	
048	4	
049	.	
050	4	
051	4	
052	8	
053	2	
054	2	
055	1	
056	6	
057	1	
058	5	
059	F2?	
060	1/X	
061	X⇄Y	
062	x	
063	RTN	
064	*LBLe	Pound mass-kilogram conversion.
065	SF2	
066	*LBLE	
067	.	
068	4	
069	5	
070	3	
071	5	
072	9	
073	2	
074	3	
075	7	
076	F2?	
077	1/X	
078	X⇄Y	
079	x	
080	RTN	
081	R/S	

REGISTERS

0	1	2	3	4	5	6	7	8	9

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9

A	B	C	D	E	I

Step	Key	Comments
001	*LBLA	
002	3	
003	2	
004	-	
005	1	°C = (°F – 32)/1.8
006	.	
007	8	
008	÷	
009	RTN	
010	*LBLa	
011	1	
012	.	°F = 1.8°C + 32
013	8	
014	x	
015	3	
016	2	
017	+	
018	RTN	
019	*LBLb	
020	SF2	British thermal unit to joule conversion.
021	*LBLB	
022	1	
023	0	
024	5	
025	5	
026	.	
027	0	
028	5	
029	5	
030	8	
031	5	
032	3	
033	F2?	
034	1/X	
035	X⇄Y	
036	x	
037	RTN	
038	*LBLc	
039	SF2	Pound per square inch to newton per square metre conversion.
040	*LBLC	
041	6	
042	8	
043	9	
044	4	
045	.	
046	7	
047	5	
048	7	
049	2	
050	F2?	
051	1/X	
052	X⇄Y	
053	x	
054	RTN	
055	*LBLd	
056	SF2	
057	*LBLD	
058	1	Pound mass per cubic foot to kilogram per cubic metre conversion.
059	6	
060	.	
061	0	
062	1	
063	8	
064	4	
065	6	
066	3	
067	F2?	
068	1/X	
069	X⇄Y	
070	x	
071	RTN	
072	*LBLe	
073	SF2	
074	*LBLE	Horsepower to watt conversion.
075	7	
076	4	
077	5	
078	.	
079	6	
080	9	
081	9	
082	9	
083	8	
084	7	
085	F2?	
086	1/X	
087	X⇄Y	
088	x	
089	RTN	
090	R/S	

LABELS

A	B	C	D	E
in-mm	ft-m	gal-l	lbf-N	lbm-kg
°F – °C	Btu-J	psi-N/m²	lb/ft³ – kg/m	hp-W
0	1	2	3	4
5	6	7	8	9

FLAGS

Flag	Use
0	
1	
2	reverse
3	

SET STATUS

FLAGS	ON	OFF	TRIG		DISP	
0	☐	☒	DEG	☒	FIX	☒
1	☐	☒	GRAD	☐	SCI	☐
2	☐	☒	RAD	☐	ENG	☐
3	☐	☒			n	2

PSEUDORANDOM NUMBER GENERATOR

Arithmetic Teacher incorporates a pseudorandom number generator. This generator supplies a sequence of numbers between zero and one which are converted into the problems displayed by the program.

The term ''Pseudorandom'' implies that the sequence of numbers is predictable from the algorithm and the initial value or seed used for the generator. A truly random device, such as a fair roulette wheel, is totally unpredictable. However, pseudorandom generators can be used to model random events provided they yield uniformly distributed numbers (i.e., as many values fall between 0.00 and 0.10 as fall between 0.10 and 0.20 etc.) and they do not repeat the same sequence of values during the simulation.

The pseudorandom number generator incorporated in *Arithmetic Teacher* is very simple but quite good. It uses the multiplicative linear congruential method:

$$u_{i+1} = \text{fractional part of } (997u_i)$$

$$\text{where } i = 0, 1, 2, \ldots$$

$$u_0 = 0.5284163^* \text{ (seed)}$$

The period of this generator has a length of 500,000 numbers and the generator passes the frequency test (chi square) for uniformity, the serial test and the run test. The most significant digits (the left hand digits) are the most random digits. The right most digits are significantly less random.

In *Arithmetic Teacher* the initial seed of .5284163 is stored at step 022. Label 5 (steps 084-096) actually generates the digits for each arithmetic problem. However, pseudorandom number generation occupies only the first six steps of label 5. These six steps and the corresponding x register contents are as follows:

STEPS	X REGISTER
LBL 5	
RCL E	old seed
9	
9	
7	997
x	seed × 997

*Other seeds may be selected but the quotient of (seed × 10^7) divided by two or five must not be an integer. Also, it would be wise to statistically test other seeds before using them.

FRC	fractional part of (seed × 997)
STO E	pseudorandom number is stored
.	to act as seed for next loop.
.	
.	

Arithmetic Teacher

Step	Key	Comment
001	*LBLa	Store initial constants and default constants.
002	0	
003	STO8	
004	2	
005	0	
006	STO7	
007	1	
008	0	
009	STOD	
010	STOE	
011	1	
012	STOA	
013	.	
014	5	
015	2	
016	8	
017	4	
018	1	
019	6	
020	3	
021	*LBLe	Store seed, either default or user.
022	STOE	
023	CLX	
024	RTN	
025	*LBLb	Input and store $n_{max}+1$. Set flag to eliminate default value.
026	SF0	
027	SPC	
028	PRTX	
029	SPC	
030	ABS	
031	1	
032	+	
033	STOD	
034	1	Calculate display setting and store for later access.
035	0	
036	x	
037	LOG	
038	INT	
039	STOA	
040	10^x	Calculate and store scale for problems.
041	STOB	
042	CLX	
043	RTN	
044	*LBLA	Select addition.
045	1	
046	GTO1	
047	*LBLB	Select subtraction.
048	2	
049	GTO1	
050	*LBLC	Select multiplication.
051	3	
052	GTO1	
053	*LBLD	Select division.
054	4	
055	*LBL1	Store +, −, x, ÷ code.
056	STOI	

Step	Key	Comment
057	SPC	Output operation code.
058	PRTX	
059	SPC	
060	*LBL9	Generate two values for a problem.
061	GSB5	
062	STOC	
063	GSB5	
064	RCLC	Generate problem.
065	GSBi	
066	RCLA	Set display.
067	X⇄I	
068	DSPi	
069	X⇄I	Scale one value.
070	R↓	
071	RCLB	
072	÷	Add values for display of x, y.
073	+	
074	0	
075	+	Place 0 in LST x.
076	RCL9	
077	X=Y?	If same problem was just given, gen new problem.
078	GTO9	
079	R↓	
080	STO9	Display problem.
081	F1?	
082	PRTX	
083	RTN	
084	*LBL5	Pseudorandom number generation.
085	RCLE	
086	9	
087	9	
088	7	
089	x	
090	FRC	
091	STOE	
092	√X	Skew numbers high.
093	RCLD	Create integer no larger than n_{max}.
094	x	
095	INT	
096	RTN	
097	*LBL1	Addition problem.
098	+	
099	STOC	
100	LSTX	
101	−	
102	LSTX	
103	RTN	
104	*LBL2	Subtraction problem.
105	STOC	
106	X⇄Y	
107	+	
108	LSTX	
109	RTN	
110	*LBL3	Multiplication problem.
111	X=0?	
112	X⇄Y	

REGISTERS

0	1	2	3	4	5	6	7	8	9
							20 − n	wrong	problem

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9

A	B	C	D	E	I
display	scale	answer	$n_{max}+1$	seed	control

Step	Key Entry	Comments
113	X=0?	
114	1	
115	x	
116	STOC	
117	LSTX	
118	+	
119	LSTX	
120	RTN	
121	*LBL4	Division problem.
122	STOC	
123	X⇄Y	
124	X=0?	
125	GSB5	
126	x	
127	LSTX	
128	RTN	
129	*LBLE	If keyboard was used to solve problem, GTO error routine.
130	LSTX	
131	X≠0?	
132	GTO7	
133	R↓	If answer is not right, display problem again.
134	RCLC	
135	X≠Y?	
136	GTO8	
137	1	Total problems done and problems wrong.
138	F2?	
139	ST+8	
140	ST-7	
141	RCL7	If 20 problems have not been done gen. another problem.
142	X≠0?	
143	GTO9	
144	SPC	Output results of lesson.
145	2	
146	0	
147	RCL8	
148	-	
149	PRTX	
150	2	
151	0	
152	PRTX	
153	÷	
154	EEX	
155	2	
156	x	
157	PRTX	
158	SPC	
159	SPC	
160	SPC	
161	SPC	
162	2	Start new lesson.
163	0	
164	STO7	
165	0	
166	STO8	
167	GTO9	
168	*LBL8	
169	SF2	Display problem again in case of error. Set wrong answer flag so that total will be incremented.
170	RCL9	
171	0	
172	+	
173	F1?	
174	SPC	
175	RTN	
176	*LBL7	Display error for cheating.
177	0	
178	÷	
179	RTN	
180	*LBL5	Undefined division patch.
181	0	
182	STOC	
183	CLX	
184	1	
185	RTN	
186	*LBLc	Print toggle.
187	F1?	
188	GTO0	
189	SF1	
190	1	
191	RTN	
192	*LBL0	
193	CF1	
194	0	
195	RTN	
196	R/S	

LABELS

A	B	C	D	E
+?	-?	x?	÷?	Answer
a Start	b (n_{max})	c P?	d	e (seed)
0 print	1 +	2 -	3 x	4 ÷
5 used	6	7 cheat	8 error	9 problem

FLAGS

Flag	Use
0	
1	Print
2	error
3	

SET STATUS

FLAGS	ON	OFF
0	☐	☒
1	☐	☒
2	☐	☒
3	☐	☒

TRIG	
DEG	☒
GRAD	☐
RAD	☐

DISP	
FIX	☒
SCI	☐
ENG	☐
n	2

Moon Rocket Lander

Step	Instruction	Comments
001	*LBLA	
002	5	
003	0	
004	0	
005	STO6	Store initial conditions.
006	5	
007	0	
008	CHS	
009	STO7	
010	6	
011	0	
012	STO8	
013	*LBL9	Divide height by 10000 for combined display of vv.0hhh
014	RCL6	
015	DSP4	
016	EEX	
017	4	
018	÷	
019	RCL7	Build vv.0hhh display taking negative values into account.
020	CF2	
021	X<0?	
022	SF2	
023	ABS	
024	+	
025	F2?	
026	CHS	
027	PSE	Display velocity and height.
028	PSE	
029	DSP0	
030	RCL8	Count down for burn.
031	PSE	
032	3	
033	PSE	
034	2	
035	PSE	
036	1	
037	PSE	
038	0	
039	PSE	Accept input.
040	*LBL5	
041	RCL8	If all fuel has been used, determine crash velocity.
042	X⇄Y	
043	X>Y?	
044	GTO2	
045	ST-8	Determine velocity and height.
046	2	
047	x	
048	5	
049	-	
050	STO9	
051	2	
052	÷	
053	RCL6	
054	+	
055	RCL7	
056	+	
057	RCL9	
058	ST+7	
059	R↓	
060	STO6	
061	INT	
062	X>0?	If no inpact go for another burn.
063	GTO9	
064	*LBL3	
065	DSP0	Flash crash velocity.
066	RCL7	
067	*LBL4	
068	PSE	
069	GTO4	
070	*LBL2	
071	RCL8	Fuel-exhausted free-fall crash velocity.
072	2	
073	.	
074	5	
075	-	
076	ST+6	
077	2	
078	x	
079	ST+7	
080	RCL6	
081	1	
082	0	
083	x	
084	RCL7	
085	X²	
086	+	
087	√X	
088	CHS	
089	GTO4	
090	*LBLB	
091	5	Flame out recovery.
092	ST-8	
093	0	
094	GTO5	
095	R/S	

REGISTERS

0	1	2	3	4	5	6	7	8	9
						x	v	Fuel	Accel.

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9

A	B	C	D	E	I

LABELS					FLAGS
A Cntrl	B Restart	C	D	E	0
a	b	c	d	e	1
0 used	1	2 fuel = 0	3 crash	4 flash	2 sign
5 restart	6	7	8	9 burn	3

SET STATUS

FLAGS	ON	OFF	TRIG		DISP	
0	☐	☒	DEG	☒	FIX	☒
1	☐	☒	GRAD	☐	SCI	☐
2	☐	☒	RAD	☐	ENG	☐
3	☐	☒			n	2

Diagnostic Program

001	*LBL0	Clear registers subroutine.	057	GSBe	
002	CLRG		058	X≤Y?	
003	P⇄S		059	GTO1	
004	CLRG		060	RTN	
005	RTN		061	*LBL1	
006	*LBLa	Function test subroutine.	062	GSBe	Decrement x.
007	RND		063	STOI	
008	RCLI		064	RCLI	I-register test.
009	X≠Y?		065	X⇄Y	
010	R/S		066	X≠Y?	
011	*LBL2		067	RTN	
012	DSZI	DSZI & RCLI subroutine.	068	GSB2	X to 0 comparisons.
013	*LBL5		069	X≠0?	
014	RCLI		070	GTO3	
015	RTN	RCLI & STOP if called	071	RTN	
016	*LBLc		072	*LBL3	
017	RCLi		073	GSB2	
018	RCLI	Verify registers & sum in R_0 subroutine.	074	X=0?	
019	X≠Y?		075	RTN	
020	R/S		076	GSB2	
021	ST+0		077	X<0?	
022	DSZI		078	RTN	
023	GTOc		079	GSB2	
024	3		080	X>0?	
025	EEX		081	GTO4	
026	2		082	RTN	
027	RCL0		083	*LBL4	Check set status on flags.
028	X≠Y?	Test R_0	084	DSZI	
029	R/S		085	F2?	
030	RTN		086	GTO5	
031	*LBLe	Decrement x subroutine.	087	DSZI	
032	1		088	F1?	
033	-		089	GTO5	
034	RTN		090	DSZI	
035	*LBLA	START & pause after first subroutine execution.	091	F3?	
036	5		092	GTO6	
037	7		093	GTO5	
038	GSB0		094	*LBL6	
039	PSE		095	DSZI	
040	GSBe		096	F0?	
041	ENT↑	Decrement x.	097	GTO7	
042	R↓		098	GTO5	
043	X⇄Y	STACK (X,Y,Z,T) TEST	099	*LBL7	
044	R↑		100	SF2	Check complement of set status on flags.
045	R↑		101	SF1	
046	X⇄Y		102	CF0	
047	R↑		103	DSZI	
048	X≠0?		104	F3?	
049	X≠Y?		105	GTO5	
050	RTN		106	DSZI	
051	GSBe	Decrement x.	107	F0?	
052	X>Y?		108	GTO5	
053	RTN		109	DSZI	
054	GSBe	X to Y comparisons	110	F2?	
055	X=Y?		111	GTO8	
056	RTN		112	GTO5	

REGISTERS

0	1	2	3	4	5	6	7	8	9
USED	USED	USED	USED	USED	USED	USED	USED	USED	USED
S0	S1	S2	S3	S4	S5	S6	S7	S8	S9
USED	USED	USED	USED	USED	USED	USED	USED	USED	USED

A	B	C	D	E	I
USED	USED	USED	USED	USED	USED

Step	Key	Comment
113	*LBL8	
114	DSZI	
115	F1?	
116	GTO9	
117	GTO5	
118	*LBL9	Check F2 for test clearing.
119	DSZI	
120	F2?	
121	GTO5	
122	GSB2	Test DEG, SIN, SIN^{-1}
123	DSP7	
124	DEG	
125	SIN	
126	SIN^{-1}	
127	GSBa	
128	COS	Test COS, COS^{-1}
129	COS^{-1}	
130	GSBa	
131	TAN	Test TAN, TAN^{-1}
132	TAN^{-1}	
133	GSBa	
134	→P	Test →P, →R
135	→R	
136	GSBa	
137	SIN	Test →HMS, HMS→
138	→HMS	
139	HMS→	
140	SIN^{-1}	
141	GSBa	
142	LOG	Test LOG, 10^x
143	10^x	
144	GSBa	
145	LN	Test LN, e^x
146	e^x	
147	GSBa	
148	√X	Test $\sqrt{X}$, X^2
149	X^2	
150	GSBa	
151	ENT↑	Test y^x, LASTx, $^1/x$
152	Y^x	
153	LSTX	
154	1/X	
155	Y^x	
156	GSBa	
157	ENT↑	Test +, −
158	+	
159	LSTX	
160	−	
161	GSBa	
162	ENT↑	Test X, ÷
163	×	
164	LSTX	
165	÷	
166	GSBa	
167	√X	Test FRC, INT
168	FRC	

Step	Key	Comment
169	LSTX	
170	INT	
171	+	
172	X^2	
173	GSBa	Test D→R, R→D
174	D→R	
175	R→D	
176	GSBa	
177	EEX	Test EEX, %
178	2	
179	X⇄Y	
180	%	
181	GSBa	
182	DSP1	Test registers 24→0 (sensitivity of higher-order registers to lower-order register changes)
183	*LBLb	
184	RCLI	
185	STOi	
186	DSZI	
187	GTOb	
188	2	
189	4	
190	X⇄I	
191	GSBc	
192	GSB0	Clear registers.
193	*LBLd	Test registers 0→24 (sensitivity of lower-order registers to higher-order register changes)
194	DSZI	
195	RCLI	
196	ABS	
197	STOi	
198	2	
199	4	
200	X≠Y?	
201	GTOd	
202	STOI	
203	GSBc	
204	9	Generate "PASS" display. −8−888888888−88
205	EEX	
206	8	
207	7	
208	1/X	
209	8	
210	CHS	
211	×	
212	SF0	Reset status for possible second pass.
213	CF1	
214	SF3	
215	RAD	
216	DSP3	Test display formatting and printing.
217	ENG	
218	PRTX	
219	SCI	
220	PRTX	
221	DSP1	
222	FIX	
223	PRTX	
224	R/S	END TEST

LABELS

A START	B	C	D	E
a Function test	b Decrementing register store	c Register check & sum	d Incrementing register store	e Decrement X
0 CL 25 REG	1 X≠Y SKIP	2 DSZI, RCLI	3 X≠0 SKIP	4 X>0 SKIP
5 RCLI & STOP	6 F3 SKIP	7 F0 SKIP	8 F2 SKIP	9 F1 SKIP

FLAGS

Flag	
0	USED
1	USED
2	USED
3	USED

SET STATUS

FLAGS	ON	OFF	TRIG		DISP	
0	☒	☐	DEG	☐	FIX	☒
1	☐	☒	GRAD	☐	SCI	☐
2	☐	☒	RAD	☒	ENG	☐
3	☒	☐			n	1

NOTES

NOTES

NOTES

HEWLETT PACKARD

1000 N.E. Circle Blvd., Corvallis, OR 97330

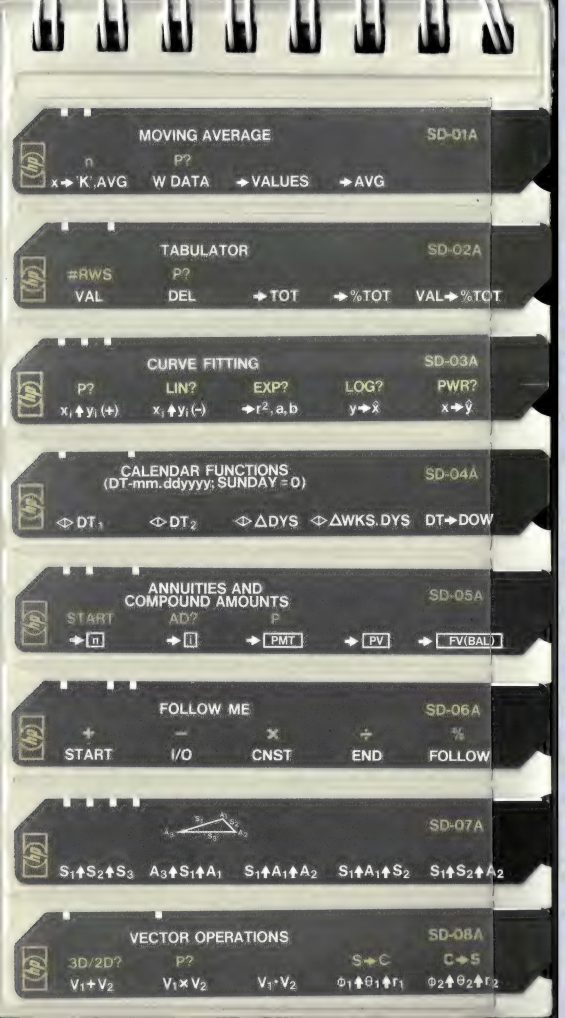
00067-90021 Rev. 11/78

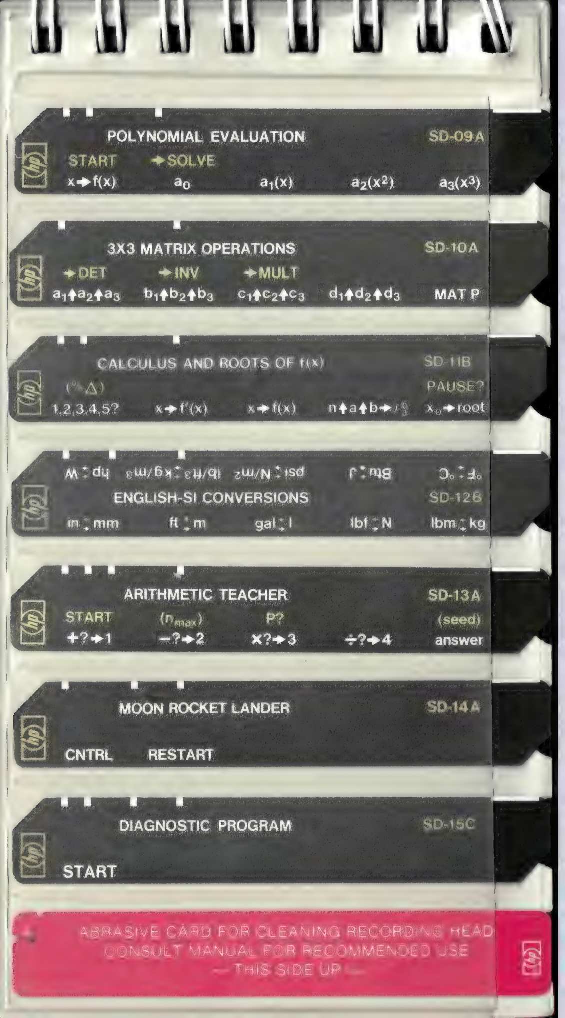
A B C • E

HEWLETT-PACKARD

HP67 HP97

Standard Pac


MOVING AVERAGE
SD-01A
n
P?
x→K, AVG
W DATA
→VALUES
→AVG
TABULATOR
SD-02A
#RWS
P?
VAL
DEL
→TOT
→%TOT
VAL→%TOT
CURVE FITTING
SD-03A
P?
LIN?
EXP?
LOG?
PWR?
$x_i \uparrow y_i$ (+)
$x_i \uparrow y_i$ (−)
$\rightarrow r^2$, a, b
$y \rightarrow \hat{x}$
$x \rightarrow \hat{y}$
CALENDAR FUNCTIONS
(DT-mm.ddyyyy; SUNDAY = 0)
SD-04A
$\Leftrightarrow DT_1$
$\Leftrightarrow DT_2$
⇔ΔDYS
⇔ΔWKS.DYS
DT→DOW
ANNUITIES AND
COMPOUND AMOUNTS
SD-05A
START
AD?
P
→ n
→ i
→ PMT
→ PV
→ FV(BAL)
FOLLOW ME
SD-06A
+
−
×
÷
%
START
I/O
CNST
END
FOLLOW
S_1 A_1 S_2 A_2 S_3 A_3
SD-07A
$S_1 \uparrow S_2 \uparrow S_3$
$A_3 \uparrow S_1 \uparrow A_1$
$S_1 \uparrow A_1 \uparrow A_2$
$S_1 \uparrow A_1 \uparrow S_2$
$S_1 \uparrow S_2 \uparrow A_2$
VECTOR OPERATIONS
SD-08A
3D/2D?
P?
S→C
C→S
$V_1 + V_2$
$V_1 \times V_2$
$V_1 \cdot V_2$
$\Phi_1 \uparrow \Theta_1 \uparrow r_1$
$\Phi_2 \uparrow \Theta_2 \uparrow r_2$


POLYNOMIAL EVALUATION
SD-09A
START
→SOLVE
x→f(x)
a0
a1(x)
a2(x²)
a3(x³)
3X3 MATRIX OPERATIONS
SD-10A
→DET
→INV
→MULT
a1↑a2↑a3
b1↑b2↑b3
c1↑c2↑c3
d1↑d2↑d3
MAT P
CALCULUS AND ROOTS OF f(x)
SD-11B
(%Δ)
PAUSE?
1,2,3,4,5?
x→f'(x)
x→f(x)
n↑a↑b→∫
x0→root
°F⇅°C
Btu⇅J
psi⇅N/m²
lb/ft³⇅kg/m³
hp⇅W
ENGLISH-SI CONVERSIONS
SD-12B
in⇅mm
ft⇅m
gal⇅l
lbf⇅N
lbm⇅kg
ARITHMETIC TEACHER
SD-13A
START
(nmax)
P?
(seed)
+?→1
−?→2
×?→3
÷?→4
answer
MOON ROCKET LANDER
SD-14A
CNTRL
RESTART
DIAGNOSTIC PROGRAM
SD-15C
START
ABRASIVE CARD FOR CLEANING RECORDING HEAD
CONSULT MANUAL FOR RECOMMENDED USE
— THIS SIDE UP —

1 2

1 2

1 2

1 2

1 2

1 2

1 2

1 2

HEWLETT hp PACKARD

A B C D E
9282-0596
MADE IN SINGAPORE